LB-OPAR: Load balanced optimized predictive and adaptive routing for cooperative UAV networks

Mohammed Gharib [a,*], Fatemeh Afghah [a], Elizabeth Serena Bentley [b]

[a] *Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, USA*
[b] *Air Force Research Laboratory Rome, NY, USA*

A R T I C L E   I N F O

A B S T R A C T

Cooperative ad-hoc UAV networks have been turning into the primary solution set for situations where establishing a communication infrastructure is not feasible. Search-and-rescue after a disaster and intelligence, surveillance, and reconnaissance (ISR) are two examples where the UAV nodes need to send their collected data cooperatively into a central decision maker unit. Recently proposed SDN-based solutions show incredible performance in managing different aspects of such networks. Alas, the routing problem for the highly dynamic UAV networks has not been addressed adequately. An optimal, reliable, and adaptive routing algorithm compatible with the SDN design and highly dynamic nature of such networks is required to improve the network performance. This paper proposes a load-balanced optimized predictive and adaptive routing (LB-OPAR), an SDN-based routing solution for cooperative UAV networks. LB-OPAR is the extension of our recently published routing algorithm (OPAR) that balances the network load and optimizes the network performance in terms of throughput, success rate, and flow completion time (FCT). We analytically model the routing problem in highly dynamic UAV network and propose a lightweight algorithmic solution to find the optimal solution with $O(|E|^2)$ time complexity where $|E|$ is the total number of network links. We exhaustively evaluate the proposed algorithm's performance using ns-3 network simulator. Results show that LB-OPAR outperforms the benchmark algorithms by 20% in FCT, by 30% in flow success rate on average, and up to 400% in throughput.[1]

1. Introduction

Cooperative unmanned aerial vehicle (UAV) networks, also known as flying ad-hoc networks (FANET), recently become extremely popular for their ease of set-up and use, low price, and high maneuverability in 3D space [1]. In such networks, nodes work cooperatively to make the communication of two far away nodes possible by acting as intermediate repeaters. Using such networks where implementing the infrastructure is not feasible, or the infrastructure is damaged, is crucially important. Wildfire monitoring [2], search and rescue after a disaster [3], ISR [4], ad hoc UAV cloud service provider [5], and border surveillance [6] are some examples to name. A promising architecture for such networks uses software-defined networking (SDN) as its cornerstone to add flexibility to the network control and management [7]. SDN separates the control plane from the data plane to pass the decision-making process to a software-based controller. There are excellent efforts in designing SDN-based topology management [7], resource management [8], path planning [9], node positioning [10],

and monitoring [11]. The SDN-based routing problem, of how to find the optimal path from the source node to the destination, is also tackled for multi-path routing [12], and low mobility UAV networks [13]. However, the SDN-based routing problem for highly dynamic UAV networks is still a challenging unresolved problem to the best of our knowledge.

In this paper, we consider a cooperative UAV network with a central controller. We assume a low-bandwidth reliable communication channel exists between the controller and the UAV nodes. However, the UAVs need to transfer a high volume of data, such as high-resolution pictures, videos, or thermal pictures. Hence, they need a high bandwidth communication channel for data transfer. In this way, the control plane and the data plane are separated, just like any other SDN-based UAV network. It is worth mentioning that the controller can be a ground or aerial controller with a line of sight or non-line of sight communication with the nodes. Such communication could be borrowed from the existing cellular network, be provided by the line-of-sight
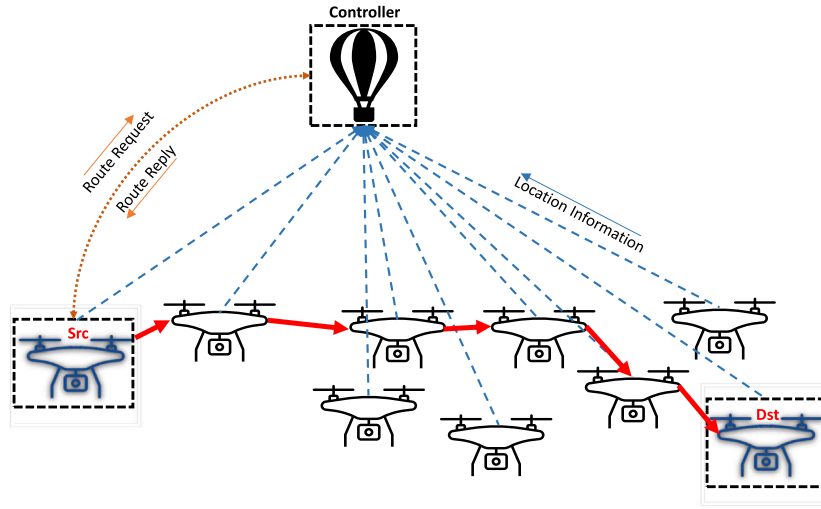

Fig. 1. System model (solid lines represent a direct link where the dashed lines represent a communication which may or may not be direct).

communication with a controller placed at a higher elevation, or use a dedicated bandwidth for command and control, as it is the common assumption in many UAV networks [1]. The controller is responsible for decision-making in all network management processes. We assume the routing, as one of the control management tasks, is also one of the controller responsibilities. Fig. 1 represents a schematic view of our system model. Since the controller is responsible for all network management tasks, any algorithm required to be performed by the controller is desired to be with a low computation and communication complexity to prevent any potential delay in network management. This fact makes the routing problem more challenging in such a network architecture.

Cooperative UAV networks typically use the conventional routing algorithms designed for the conventional ad-hoc networks such as Mobile Ad-hoc NETwork (MANET) and Vehicular Ad-hoc NETwork (VANET). Ad-hoc On-demand Distance Vector (AODV) [14], Destination Sequenced Distance Vector (DSDV) [15], and Optimized Link State Routing (OLSR) [16] are some instances. Such algorithms aim at finding the shortest path between the source and the destination and do not consider the other network characteristics. Furthermore, the highly dynamic nature of UAV networks leads to fast disconnection and consequently a need for multiple reroute processes [17]. Multiple reroute processes dramatically increase the routing communication complexity as well as overall end-to-end communication delay. In this case, the data packets belonging to the same data file/stream may traverse different paths, and hence they arrive out-of-order. The same side effect is tackling the multi-path routing algorithms. TCP deals with some of out-of-order packets as the lost packets and orders retransmission.

We recently published an optimized predictive and adaptive routing (OPAR) [18] for cooperative UAV networks to consider the path lifetime in addition to the path length in the path selection process. The aim was to reduce the reroute process's overheads. We find that considering the path lifetime can significantly improve UAV networks' performance in terms of throughput, flow completion time (FCT), and flow success rate. We further find that considering only the path lifetime and path length in selecting the paths may cause an unbalanced network load. Since the network uses a shared medium, unbalanced load in the network may cause performance degradation in throughput and consequently FCT, especially in the highly loaded networks. Accordingly, in this paper, we extend the OPAR algorithm to propose a load-balanced OPAR (LB-OPAR) routing algorithm by considering the network load in addition to the path lifetime and path length in selecting the paths. We show that this consideration does not impose any extra routing traffic to the network in comparison with OPAR, in the SDN-based setting. Accordingly, we model the entire problem of the SDN-based routing algorithm in a cooperative UAV network

with an analytical optimization model. The proposed model is binary linear programming (BLP) problem. While BLP problems are very well-known to be NP-complete problems, we show that in the specific case of the modeled BLP, we can find the optimized path using a graph-based algorithmic solution. The algorithmic solution is lightweight with the computational complexity of $O(|E^2|)$ where $|E|$ is the number of network links.

LB-OPAR needs only small-size messages to be sent to the controller periodically. Each message contains three consequent positions of the UAV node. The controller predicts the link lifetime of the UAV nodes and uses the predicted values in solving the optimization problem. The prediction algorithm is lightweight, with a computation complexity of $O(|V|^2)$ where $|V|$ is the number of UAV nodes. The controller uses its own data to calculate the communication load on each node. Hence, no further communication is required. The low communication complexity and the low computational complexity of the LB-OPAR algorithm make it a perfect fit for the SDN-based model.

To evaluate the performance of LB-OPAR, we use the ns-3 network simulator [19]. We simulate the network to compare LB-OPAR with the well-known baseline routing algorithms AODV, DSDV, and OLSR as benchmarks. We compare the results with those of OPAR as well. We show in Section 2 that the state-of-the-art uses these benchmarks to find the routes in the cooperative UAV networks or build new routing algorithms on the basis of these benchmarks. We selected the algorithms for comparison that cover the wide range of reactive and proactive algorithms as well as distance vector and link-state ones. We measure the network throughput, FCT, and flow success rate as the performance metrics. We show that LB-OPAR improves the network throughput by up to four times the benchmarks. It decreases the FCT for average 20% and increases the flow success rate by about 30%.

This work's main contribution is to propose a load-balanced routing algorithm for SDN-based cooperative ad-hoc networks. More specifically, (*i*) it analytically models the routing problem in the mentioned systems with linear optimization technique and considers the path length, lifetime, and load altogether; (*ii*) it proposes an algorithmic solution for the proposed problem; (*iii*) it analytically proves the solution's complexity; (*iv*) it proposes a lightweight highly accurate link lifetime prediction method; (*v*) it exhaustively evaluates the performance of the proposed algorithm and compares it with the benchmark solutions.

The rest of this paper is organized as follows. The state-of-the-art is reviewed in Section 2. We propose the analytical model with its theoretical solution of the LB-OPAR in Section 3. The performance of LB-OPAR is evaluated and compared to the baseline solutions in Section 4. Finally, we conclude the paper in Section 5.

2. Related work

Cooperative UAV networks typically use conventional routing algorithms, which are proposed originally to be used in mobile ad-hoc networks (MANETs), vehicular ad-hoc networks (VANETs), or wireless sensor networks (WSNs). However, there are several routing algorithms dedicatedly proposed for cooperative UAV networks. Thus, in this section, we first review the conventional routing algorithms, then we present the algorithms proposed dedicatedly for UAV networks. Finally, we review the proposals for SDN-based UAV network routing algorithms to cover the state-of-the-art.

Generally, there are two categorizations for conventional ad-hoc routing algorithms [20]. Each of them looks at the problem from a different viewpoint. However, they all are proposed to find the path with the least number of hop counts, and they do not consider other metrics such as path quality, reliability, or lifetime. The first looks at the algorithms based on their nature of being proactive, reactive, or hybrid. Proactive routing protocols periodically keep track of the network topology changes, regardless of the demand for new routes. Optimized link-state routing protocol (OLSR) [16] and destination sequence distance vector (DSDV) [15] are two well-known instances of this category. In contrast to proactive protocols, reactive protocols attempt to find a route only when a new route is needed. Ad-hoc on-demand distance vector (AODV) [14] is a very well-known instance of this category. The second categorization considers the method of implementing the routing algorithms to be link state or distance vector. In the link-state approach, each node builds a general view of the entire network topology to find the optimized paths. The OLSR protocol belongs to this category. In contrast, each node needs to keep only the next hop toward the destination and the path cost in the distance-vector approach. Both AODV and DSDV use this approach in their routing.

Most of the routing algorithms proposed dedicatedly for UAV networks are built based on the conventional ad-hoc routing algorithms and extend them to consider another metric in their path selection process. As an instance, Bomio et al. [21] proposed enhanced AODV (E-AODV) by taking into account a route reliability criterion. They considered a 2D area to check whether two nodes are getting closer or not. If their distance is going to be increased, the corresponding link is discarded. Li et al. [22] proposed link-stability estimation-based pre-emptive routing (LEPR) based on AODV. LEPR finds multiple disjoint paths using the AODV algorithm, calculates the expected transmission count (ETX) metric of [23] for each link of the route, and keeps all routes to use them when the path with the best ETX becomes not available anymore.

Pu et al. [24] proposed link quality and traffic-load aware OLSR (LTA-OLSR) to find the path with the highest quality from the set of the shortest paths found by OLSR. Song et al. [25] proposed OLSR-based mobility and delay prediction (OLSR-PMD) to choose the more stable next-hop nodes based on a prediction algorithm. Rosati et al. [26] proposed predictive OLSR (P-OLSR) to add weight to the ETX metric of each link based on the corresponding nodes' relative speeds. Lyu et al. [27] proposed Q-Network Enhanced Geographic Ad-Hoc Routing Protocol based on GPSR (QN-GPSR). Authors reduced the premier forwarding of GPSR [28] algorithm by applying Q-learning, a reinforcement learning algorithm, into the GPSR routing protocol. Their results show that QN-GPSR has a higher packet delivery ratio and a lower end-to-end delay than the basic GPSR protocol. However, applying a Q-network increases the computational complexity of the GPSR algorithm by order of magnitude. Liu et al. [29] proposed another Q-learning based Routing protocol for Flying Ad Hoc Networks and named it Q-learning based multi-objective routing (QMR). Authors show that QMR provides a higher packet delivery ratio, lower delay, and energy consumption than existing Q-learning-based routing protocols. However, the vast imposed computational complexity by the Q-learning algorithm is the main drawback of this algorithm. Adaptive and Reliable Routing Protocol with Deep Reinforcement Learning (ARdeep) [30] is also proposed to be applied to UAV networks. Although the authors show that it performs better than the basic GPSR algorithm in terms of average end-to-end delay and packet delivery ratio, it increases the computational complexity by order of magnitude, just like other deep learning-based algorithms. Although most of the computational complexity source of ML algorithms belongs to the training phase, a learned model might not be able to lead to high precision results in a network with a highly dynamic nature [31]. To address this shortcoming, the learning-based methods keep the learning phase active during the network operation time to be regularly adjusted with the new live data where such online learning needs high computational complexity.

Several other routing algorithms are proposed dedicatedly for UAV networks, but they are special-purpose algorithms or only applicable to the networks with specific missions. For instance, Sunder et al. [32] designed a routing protocol for a UAV-based fueling system. Coelho et al. [33] designed a routing algorithm for UAV networks where UAVs are used as WiFi repeaters to maximize the coverage. The work of [34] designed for package delivery networks. Cheriguene et al. [35] proposed a Swarm energy-efficient multicast routing protocol for UAVs flying in group formations, to facilitate the control and information delivery among the UAVs while minimizing inter-UAV packet loss, packet re-transmission, and end-to-end delay. It is obvious that such mission-specific routing algorithms cannot be used as general-purpose routing for all cooperative UAV networks.

An SDN architecture is proposed to be used in UAV networks to increase the network control flexibility and improves the network performance. Different architectures are proposed for SDN-based UAV networks [36–39]. All the proposed architectures have a controller or a set of distributed controllers that has/have direct, reliable communication with the UAV nodes. Using such an architecture, any of the network management tasks could be migrated into the controller. However, the migrated tasks do not impose huge computational or communication complexity into the controller. Routing, as an essential network management task, could be migrated into the controller. To the best of our knowledge, the problem of finding an optimal path in an SDN-based UAV network has not yet been substantially studied. Several recently published works mentioned this problem [12,13,39].

Ramaprasath et al. [13] proposed an SDN-based routing algorithm that utilizes AODV in the initialization phase of the network. Alas, this proposal cannot capture the high dynamicity of the UAV networks due to its multiple long wait times during the routing procedure. Hence, this algorithm's networks of interest are the UAV networks with deficient mobility. Authors of [12] proposed an SDN-based routing algorithm that finds all possible paths toward the destination and divides the entire data file among them. Each part of the data file is then sent through one of the paths. As the out-of-order packet arrival is one of the main challenges of the highly dynamic networks, this works increases the number of such packets by using multiple paths. Authors also assume that each UAV is equipped with different network interfaces. Hence, it can send the data file part from different paths concurrently, which poses a hardware limitation for networks aimed at using this algorithm. Qi et al. [39] proposed traffic differentiated routing (TDR), a new SDN architecture for UAV networks that includes two types of controllers, cluster controllers and a coordination controller. In this architecture, the network is clustered, and each cluster needs a controller. To arrange different cluster controllers, a coordination controller is also required. The routing algorithm of this paper is proposed to find the routes inside the clusters. Besides their new network architecture, authors assumed that the controller is aware of the current location and the nodes' speed. Using such information, the authors proposed an optimization problem in which finding the optimal solution is not practically feasible. Hence, they use approximation methods to find a near-optimal path. As we mentioned earlier, the routing problem in SDN-based cooperative UAV networks needs substantial work. Accordingly, we propose LB-OPAR as a general-purpose routing algorithm for SDN-based UAV networks in this paper. Table 1 represents a comparison between routing algorithms proposed dedicated for UAV networks or could be used in them, including LB-OPAR.

Table 1

Routing Algorithms for Cooperative UAV Networks. (The comparison of this table is partially based on the results of Section 4 of this paper.).

Algorithm	Mobility Support	Routing Metric	Commun. Overhead	Comput. Overhead	SDN-based/ Fully-dist.	Load Balancing	Success Rate
AODV [14]	Good	Path len.	High	Low	Fully-dist.	×	Medium
DSDV [15]	VLimited	Path len.	VHigh	Low	Fully-dist.	×	VLow
OLSR [16]	VLimited	Path len.	VHigh	Low	Fully-dist.	×	VLow
E-AODV [21]	Good	Distance	High	Low	Fully-dist.	×	Low
LEPR [22]	Good	ETX	High	Low	Fully-dist.	×	Medium
LTA-OLSR [24]	VLimited	RSSI+load	VHigh	Low	Fully-dist.	✓	VLow
OLSR-PMD [25]	VLimited	Delay	VHigh	Low	Fully-dist.	×	VLow
P-OLSR [26]	VLimited	Weighted ETX	VHigh	Low	Fully-dist.	×	VLow
QN-GPSR [27]	Limited	Link Quality	VHigh	VHigh	SDN-based	×	Low
ARdeep [30]	Medium	PER[a]	—	VHigh	SDN-based	×	—
QMR [29]	Limited	Delay & Energy	VHigh	VHigh	SDN-based	×	Low
[12]	VLimited	Multipath	VHigh	Low	SDN-based	×	Low
[13]	VLimited	Path Len.	High	VHigh	SDN-based	×	VLow
TDR [39]	VLimited	Link Availability	—	VHigh	SDN-based	×	—
OPAR [18]	VGood	Path len. & lifetime	Low	VLow	SDN-based	×	VHigh
LB-OPAR	VGood	Path len., load & lifetime	VLow	Low	SDN-based	✓	VHigh

[a] Packet error rate.

3. Proposed algorithm

We model the routing problem in SDN-based cooperative UAV networks with a linear programming (LP) optimization model. The optimization model considers path length, path lifetime, and network load altogether. We define the path lifetime as the minimum lifetime of all the links forming the path. Hence, we need to know the link lifetime of each network link. Obviously, the link lifetime depends mostly on the nodes' movement. Hence, we propose a link lifetime prediction algorithm to support LB-OPAR. We further consider the path load as the maximum node load for the nodes participating in the path. We define the node load as the number of the neighboring nodes engaged with data transmission, affecting the medium access accessible with the corresponding node. In other words, the node load is the number of nodes, in the corresponding node's adjacency, that are sending or forwarding data packets, i.e. they are a source of a data flow or engaged as forwarding nodes in a communication flow. Since the controller manages the routes and knows the nodes' location, it can calculate nodes' load. The proposed optimization problem variables are Binary, which means that the model is a Binary LP problem. The Binary LP problems are well-known to be non-polynomial problems in the NP-complete category [40]. However, we show that the proposed problem is an especial case in which its solution could be found using the graph-based algorithmic method.

In this section, we propose the LP problem, considering that we know the links' lifetime and nodes' load. We then propose the graph-based algorithmic solution, which can find the solution of the Binary LP problem. Finally, we present the details of how to predict the link lifetime and calculate the node load by the controller. Table 2 represents the notations used throughout this paper.

3.1. LB-OPAR: SDN-Based load balanced Optimized Routing

On the one hand, it is generally a challenging problem to consider multiple metrics in defining an optimization problem. Multi-objective linear programming (MOLP) is a trivial modeling technique in such cases. MOLP solution, if feasible, leads in a set of all efficient extreme points, i.e. all maximal efficient faces. The optimal solution, then, is based on the weights of importance of different objective functions. On the other hand, finding a path requires an optimization problem with integer or Binary variables, where solving the optimization problems, in most cases, results in non-integer solutions. Rounding the non-integer solutions to the closest integer value leads to a non-optimal solution. Finding the optimized solution of most of the proposed optimization

Table 2

Table of notations.

n	Number of UAVs
u_i	The ith UAV
$G(V, E)$	Network Graph
V	Set of graph vertices
E	Set of graph edges
$e(i, j)$	The edge from node u_i to node u_j
$\mathcal{T}$	Link lifetime prediction matrix
$\mathcal{L}$	The vector of nods' load.
$\tau_{(i,j)}$	Link lifetime of $e(i, j)$
l_i	The load of node u_i
$x_{(i,j)}$	Binary variable corresponding to $e(i, j)$
T	The inverse of path lifetime
w_1	Path length weight
w_2	Path lifetime weight
w_3	Path load weight
ς	The matrix of all lifetime-load costs
$\varsigma_{(i,j)}$	the lifetime-load cost of $e_{(i,j)}$
ς_{path}	The maximum lifetime-load cost of a path
$x_i(t_0)$	The position of u_i at time t_0 along x-axis in 3D space
$y_i(t_0)$	The position of u_i at time t_0 along y-axis in 3D space
$z_i(t_0)$	The position of u_i at time t_0 along z-axis in 3D space
α_i	The azimuthal degree of node u_i direction
θ_i	The polar degree of node u_i direction
$v_i(t_0)$	The speed of node i in time t_0
$a_i(t_0)$	The acceleration of node i in time t_0
$d_{(t_0)}^{(i,j)}$	Euclidean distance between u_i and u_j at time t_0
$d_{(t_0,t_1)}^{(i)}$	Euclidean distance traversed by node u_i in time $[t_0 \quad t_1]$
$\mathfrak{R}$	Communication range

problems in routing and path planning is not feasible [32,34,39]. Accordingly, the optimization problem's design has to be such that finding the optimal solution becomes feasible, if possible. Hence, we carefully designed the following optimization problem, which is a Binary LP problem. In this problem, we assume that we have a matrix of all links' lifetimes, i.e. $\mathcal{T}$, as well as the vector of all nodes' load, i.e. $\mathcal{L}$.

$$\min \sum_{\substack{(i,j)\in E \\ i\in\{1,\ldots,n\} \\ j\in\{1,\ldots,n\}}} w_1 x_{(i,j)} + w_2 T + w_3 L$$

Subject To:
$$\sum_{(s,i)\in E} x_{(s,i)} = 1 \tag{1}$$

$$\sum_{(i,s)\in E} x_{(i,s)} = 0 \tag{2}$$

$$\sum_{(i,d)\in E} x_{(i,d)} = 1 \tag{3}$$

$$\sum_{(d,i)\in E} x_{(d,i)} = 0 \tag{4}$$

$$\sum_{\substack{(i,j)\in E \\ i\neq s}} x_{(i,j)} = \sum_{\substack{(j,k)\in E \\ j\neq d}} x_{(j,k)} \tag{5}$$

$$T \geq \frac{x_{(i,j)}}{\tau_{(i,j)}} \tag{6}$$

$$x_i = \sum_{\substack{j=0 \\ j\neq i}}^{n} x_{(i,j)} \tag{7}$$

$$L \geq x_i l_i \tag{8}$$

$$x_{(i,j)} \in \{0,1\} \tag{9}$$

$$0 \leq T \leq 1 \tag{10}$$

The objective function of this problem is to minimize the path length as well as path load, and maximize the path lifetime. The significance of each of the path length, lifetime, and load is set by setting the weights w_1, w_2, and w_3, respectively, where $\sum_{i=1}^{3} w_i = 1$. The Binary variables $x_{(i,j)}$s represent whether the corresponding network link $e_{(i,j)}$ participate in the optimal path or not. If the link $e_{(i,j)}$ is selected to be a part of an optimized path, $x_{(i,j)}$ is set to one, elsewhere it is set to zero. In this objective function, T and L represent the inverse path lifetime, and the path load, respectively. While T has to be minimized according to the objective function, it has to be greater or equal to $\frac{x_{(i,j)}}{\tau_{(i,j)}}$ for all links $x_{(i,j)}$s, according to Constraint (6). It is the idea to find the path lifetime, i.e. the minimum link lifetime for the links participated in the path, and consider it in the objective function. Similarly, the path load L has to be minimized. To find the path load we minimize L in the objective function where in Constraint (8) we force L to be greater than or equal to $x_i l_i$, where according to Constraint (7) $x_i = \sum_{j=0, j\neq i}^{n} x_{(i,j)}$, and l_i is the network load of node i.

We need Constraints (1) and (2) to let the source node start the path and prevent it from being in a loop. Similarly, we need Constraints (3) and (4) to let the destination node be chosen as the last node in the path and terminate the path selecting operation. Constraint (5) assures that the intermediate node, which receives an active link, has to have an outgoing active link. As we discussed earlier, the proposed problem variables are Binary, which is shown in Constraint (9). Finally, we need Constraint (10) in practice to remove the links with a very short lifetime. How short the link could be to be discarded depends on the metric chosen for link lifetime to be second, millisecond, etc.

3.2. Optimization problem solution

Generally, Binary LP problems are in the NP-complete category [40], and hence, they do not have a polynomial-time solutions. In this section, we show that the LB-OPAR problem is a special case of Binary LP problems where the optimal solution can be found via graph-based algorithmic solution. Needless to say, the proposed algorithmic solution might not be applicable to other Binary LP problems. While the solution is as follows, Algorithm 1 represents the pseudocode of the algorithm. We prove through Lemma 1 that the output of Algorithm 1 is the optimal solution of the Binary LP problem proposed in Section 3.1. The computation and space complexity of Algorithm 1 is also investigated in Lemma 2.

To find the solution to the proposed optimization problem, i.e. the optimal path, we consider the network as a graph $G(V, E)$ where V and E are the sets of graph vertices and edges, respectively. We consider any UAV node as a vertex in graph G. Any two neighbor UAVs which are located at the communication range of one another are shown by an edge in graph G. Thus, the edge $e_{(i,j)}$ in graph G represents a physical link between UAV nodes u_i and u_j. If the communication range between two nodes is not identical, we can consider $e_{(i,j)}$ as a directional edge starting from node u_i toward node u_j, without loss of generality.

To find the optimal path, we first define a lifetime-load cost metric for each edge. We name this metric $\varsigma_{(i,j)}$ for edge $e_{(i,j)}$ and we can

---

ALGORITHM 1: Finding the solution of the optimization problem

$objValue = \infty$, {Set the initial objective value equal to ∞.}
$path = \emptyset$, {Set the initial path to empty path.}
$\varsigma \leftarrow w_2 \frac{1}{\mathcal{T}} + w_3 \mathcal{L}$, {Form ς, a $n \times n$ matrix containing the lifetime-load cost of all graph edges. The lifetime-load cost for each edge $e_{(i,j)}$ is calculated as $\varsigma_{(i,j)} = w_2 \frac{1}{\tau_{(i,j)}} + w_3 l_j$. }
$\varsigma_{Sorted} = descentSort(\varsigma)$, {Sort the lifetime-load cost matrix in descending order.}
$G'(V, E) = G(V, E)$, {Let $G'(V, E)$ equal to the main graph $G(V, E)$.}
while $isPath(G'(V, E), src, dst)$ **do**
   $newPath = BFS(G'(V, E), src, dst)$, {To find the shortest path with BFS algorithm.}
   if $newObjValue < objValue$ **then**
     $path = newPath$, {Set the path equal to the new path.}
     $objValue = newObjValue$, {Set the objective value equal to the objective value of the new path.}
   end if
   $\varsigma_{path} \leftarrow lifetimeLoadCost(newPath)$, {Calculate the maximum path lifetime-load cost for the edges forming the $newPath$.}
   $G'(V, E) \leftarrow remove(\varsigma_{path}, \varsigma_{Sorted}, G'(V, E))$, {Remove all links with the lifetime-load cost greater than or equal to the ς_{path}.}
end while

---

calculate this metric as $\varsigma_{(i,j)} = w_2 \frac{1}{\tau_{(i,j)}} + w_3 l_j$. The basic idea of Algorithm 1 is to use the breadth-first search (BFS) algorithm [41] to find the shortest path in graph $G(V, E)$, regardless of its lifetime or load. BFS is the algorithm with the lowest computational complexity to find the shortest path in a graph. We then calculate the objective value for the path and the lifetime-load cost, i.e. $\varsigma_{(i,j)}$, of all edges forming the path. Next, we remove all graph edges with less or equal lifetime-load cost than the maximum $\varsigma_{(i,j)}$ of the path. We apply the BFS algorithm again on the new graph. If the new path has a lower objective value, we consider it as the optimal path; elsewhere, we discard the new path. The procedure is repeated until no more paths stay between the source and the destination nodes. Algorithm 1 shows the detailed pseudocode of the proposed algorithm. We show in Lemma 1 that the output of Algorithm 1 is the optimal path. We then calculate the space and computational complexity of this algorithm in Lemma 2.

Lemma 1. *If there is more than one path from the source node toward the destination, Algorithm 1 returns the optimal path.*

Proof. We prove this lemma using proof by induction. If there is at least one path, we know that the BFS algorithm will find it at the first iteration of the algorithm, and it is the optimal path. Assume that the algorithm runs $(n+1)$ iterations to return the final answer. Again, assume that after nth iteration, path p is returned as the optimal path, and it is optimal until then. To prove the lemma, we need to show that the output of the last iteration, i.e. $(n+1)^{th}$ iteration, will be optimal, too. In this case, we have to show that the link(s) that we removed from the graph in nth iteration did not lead to discarding the optimal path. We use proof by contradiction to prove it. Hence, we assume that there exists a path $\bar{p}$ with an objective value less than that of p, which at least one of its links is discarded in nth iteration. It is worth mentioning that, since Algorithm 1 first finds the shortest path and then calculates the lifetime-load cost and discards some links, $\bar{p}$ cannot be shorter than p. If one of $\bar{p}$ links is discarded, it means that the link has a lifetime-load cost more than or equal to the maximum lifetime-load cost of p's links. When $\bar{p}$ is not shorter than p, all of its links have to have lifetime-load costs less than that of the maximum cost of p's links to become the optimal path. We meet a contradiction; hence, the lemma is proved. If there exists more than one path from the source node toward the destination, Algorithm 1 returns the optimal path. $\square$

Lemma 2. *The computational complexity of Algorithm 1 is $O(|E|^2)$ for connected networks, i.e. there is at least one path between any pair of nodes. This algorithm's space complexity is also $O(|V| + |E|)$.*

Proof. We know that the computational complexity of the BFS algorithm is $O(|V| + |E|)$ where $|V|$ and $|E|$ stand for the number of graph vertices and edges, respectively [41]. In Algorithm 1, in the worst case, we need to perform BFS $|E|$ times. We further need to sort the lifetimes-load cost matrix. The worst-case complexity of sorting the lifetime-load cost matrix is $O(|E| \log |E|)$. Hence, the entire algorithm complexity is $O(|E|(|V| + |E|) + |E| \log |E|) = O(|E|^2 + |E||V| + |E| \log |E|)$. We know that $O(|E|) \geq O(|V|)$ for the connected networks. Hence, the computational complexity of Algorithm 1 is $O(|E|^2)$. We further know that the space complexity of the BFS algorithm is $O(|V|)$. We need further $O(|E|)$ space to store the lifetime-load cost matrix. The space complexity of the sort is $O(1)$. Hence, the space complexity of Algorithm 1 is $O(|V| + |E|)$. □

It is worthy to mention that we have two reasons in choosing our LP-based methodology instead of workflow scheduling techniques such as the reference one multi-objective heterogeneous earliest-finish-time (MOHEFT) [42], and the much improved one Fast workflow scheduling approach based MOHEFT using BAcktracking and CHeckpointing (FAMOBACH) [43]. Workflow scheduling techniques, in general, work with much lower computational complexity than LP algorithms and lead to near-optimal solutions. However, our proposed solution finds the exact optimal answer to the problem as we proved it in Lemma 1, and the computational complexity of our solution is $O(|E|^2)$ at its worst case, as we shown in Lemma 2, which is fairly low in comparison with the mentioned alternatives.

3.3. Link lifetime prediction and node load calculation

The optimization problem of Section 3.1 considers a $n \times n$ matrix $\mathcal{T}$ which contains link lifetimes as well as vector $\mathcal{L}$ with n elements which contain nodes' load. In this section, we describe how we predict the link lifetimes and calculate node load to form $\mathcal{T}$ and $\mathcal{L}$. We start with $\mathcal{T}$ where we need only three consecutive node locations to fill it. Each node regularly sends a packet to the controller, including its last three consecutive locations in 3D space. The controller, in its turn, calculates the azimuthal and polar directions of each node as well as their speed and acceleration. Using this information, the controller calculates the time for each pair of neighboring nodes that they go out of the communication range of each other. To use simpler notations and without the loss of generality, we consider the communication range of all nodes equal and represent it with $\mathfrak{R}$. It is worthy to mention that our predicted lifetimes stay precise if the corresponding nodes do not change their direction or acceleration.

We represent the location of node u_i at time t by the tuple $(x_i(t), y_i(t), z_i(t))$. We show the distance traversed by the node u_i in the time interval $[t_1 \quad t_2]$ by $d^{(i)}_{(t_1, t_2)}$ and it could be easily calculated by Eq. (11).

$$d^{(i)}_{(t_1, t_2)} = \sqrt{[x_i(t_2) - x_i(t_1)]^2 + [y_i(t_2) - y_i(t_1)]^2 + [z_i(t_2) - z_i(t_1)]^2} \quad (11)$$

Considering the three consecutive node locations at times t_0, t_1, and t_2, we calculate the azimuthal angle α_i and polar angle θ_i of the node's movement direction using Eqs. (12) and (13), respectively.

$$\alpha_i = tan^{-1}\left(\frac{y_i(t_2) - y_i(t_1)}{x_i(t_2) - x_i(t_1)}\right) \quad (12)$$

$$\theta_i = cos^{-1}\left(\frac{z_i(t_2) - z_i(t_1)}{d^{(i)}_{(t_1, t_2)}}\right) \quad (13)$$

$$= tan^{-1}\left(\frac{\sqrt{[x_i(t_2) - x_i(t_1)]^2 + [y_i(t_2) - y_i(t_1)]^2}}{z_i(t_2) - z_i(t_1)}\right)$$

We can calculate speed v_i and acceleration a_i of node u_i respectively using Eqs. (14) and (15).

$$v_i(t_1) = \frac{\sqrt{d^{(i)}_{(t_0, t_1)}}}{t_1 - t_0} \quad (14)$$

$$a_i(t_2) = \frac{v_i(t_2) - v_i(t_1)}{t_2 - t_0} \quad (15)$$

Next, we can calculate the predicted location of the node u_i at time $(t_2 + \Delta t)$ using Eq. (16). Then, we can calculate the distance between nodes u_i and u_j at time $(t_2 + \Delta t)$ using Eq. (17). We represent this distance by $d_{(i,j)}(t_2 + \Delta t)$ where Δt is unknown. Knowing the communication range of nodes, i.e. $\mathfrak{R}$, we can calculate the time Δt using Eq. (18). This variable is the time required for the nodes to go out of the communication range of one another and represents the lifetime of the link $e_{(i,j)}$. Eq. (18) is a simple nonlinear equation with only one unknown variable, i.e. Δt. This equation can be easily solved with any lightweight numerical method such as the Bisection method. It is worth mentioning that the LB-OPAR is a modular routing algorithm. Hence, the prediction algorithm proposed here, while its accuracy is validated in Section 4.2, could be replaced with any other prediction algorithm with no need for any modification in the overall routing solution.

$$\begin{cases} x_i(t_2 + \Delta t) = x_i(t_2) + (v_i(t_2)\Delta t + \frac{1}{2}a_i\Delta t^2)sin(\theta_i)cos(\alpha_i) \\ y_i(t_2 + \Delta t) = y_i(t_2) + (v_i(t_2)\Delta t + \frac{1}{2}a_i\Delta t^2)sin(\theta_i)sin(\alpha_i) \\ z_i(t_2 + \Delta t) = z_i(t_2) + (v_i(t_2)\Delta t + \frac{1}{2}a_i\Delta t^2)cos(\theta_i) \end{cases} \quad (16)$$

$$d^{(i,j)}_{(t_2 + \Delta t)} = \sqrt{\begin{array}{l}(x_i(t_2 + \Delta t) - x_j(t_2 + \Delta t))^2 + \\ (y_i(t_2 + \Delta t) - y_j(t_2 + \Delta t))^2 + \\ (z_i(t_2 + \Delta t) - z_j(t_2 + \Delta t))^2\end{array}} \quad (17)$$

$$d^{(i,j)}_{(t_2 + \Delta t)} - \mathfrak{R} = 0 \quad (18)$$

In a wireless network, a node can start transmitting only when the channel is empty, which happens when no other node in the communication range of the sending node transmits a packet. If, for any reason, more than one node attempt to send a packet simultaneously, a collision will happen. Generally, wireless networks, via their medium access control (MAC) protocols, try to avoid the collision by preventing one of the nodes from sending its own packets at the same time. This fact leads to a delay in flow completion time. In our SDN-based system, the controller is aware of all transmitting flows and their paths. Hence, it can calculate the number of nodes that may attempt to send a packet in the communication range of a sending node. We refer to this number as the node's load.

To calculate each node's load, the controller starts the network operation with all zeros for the nodes' load. Any time the controller receives a route request, it calculates the optimal path and determines the nodes affected by the new route. Any node included in the new route and all its one hop neighbors are considered as the nodes affected by the new route. Since the controller potentially knows all nodes' locations, it can also calculate the distance between each pair of nodes, and it further knows the communication range of each node. Hence the controller can determine the affected nodes by the new route. The controller then increases the load of the determined nodes by one. Every time that a route becomes unavailable, the controller decreases the affected nodes' load by one. In two cases, a route is considered unavailable. First, the source node does not receive the TCP acknowledgment packets, and consequently, the source node orders a reroute. Second, the source node receives the acknowledgment of the last packet of the file and sends a flow termination message to the controller.

4. Performance evaluation

To evaluate the performance of LB-OPAR, we use ns-3 network simulator. Ns-3 is a well-known discrete-event simulator widely used

Table 3
Simulation setting.

Simulator version	ns-3 3.30
Number of UAVs	[50 100]
Area size	300 × 2000 × 50 m
Transmission power	7.5 dB m
Number of concurrent flows	[1 10]
File size	5 MB
Simulation time	500 s
Speed range	[0 50] m/s
Mobility models	3D RWP
Traffic type	TCP NewReno
Wireless communication standard	IEEE 802.11b
Propagation loss model	Free-space propagation loss
Propagation delay model	Constant speed propagation delay

for research and development [19]. We simulate the network for different scenarios with a different number of nodes and a different number of concurrent file-transfer flows to exhaustively evaluate the network performance in different situations. Network throughput, flow completion time, and flow success rate are measured as the performance evaluation metrics. To show the significant improvement of LB-OPAR in the network performance, we compare its results with the baseline benchmarks AODV, OLSR, DSDV as well as the OPAR algorithm. We select the baseline algorithms that cover the proactive and reactive, link-state and distance-vector, as well as SDN-based and fully distributed routing algorithms. In this section, we first describe the simulation setting. Next, we evaluate the prediction algorithm proposed in Section 3.3. Since LB-OPAR is sensitive to its weights w_1, w_2, and w_3, we then perform an analysis on these weights to find the proper value for them in different network settings. Finally, we represent the results of the comparison among LB-OPAR, AODV, DSDV, OLSR, and OPAR, along with the analysis.

4.1. Simulation setting

We simulate a UAV network with n nodes in a 2000 × 300 × 50 3D space. We perform two sets of simulation scenarios. In the first set of the simulation scenarios, we vary the number of nodes from 50 to 100 in an increment step of 5. In this set of scenarios, five source nodes start sending their data to five corresponding destination UAVs. The generated traffic is FTP over TCP NewReno. A 5 MB data file is sent in each flow. In the second set of scenarios, we set the number of nodes equal to fifty, i.e. $n = 50$, and vary the number of concurrent flows from one to ten. Each simulation instance is run for 500 s. UAV nodes move based on two different random movement patterns, 3D random waypoint, and 3D Gauss–Markov model. While random waypoint chooses its waypoints in a completely random manner, the Gauss–Markov model is closer to pre-planned scenarios. It prevents the sudden significant direction change by applying a limitation on the variability of the movement angle. We choose these two mobility models to show that LB-OPAR shows excellent performance in both random and pre-planned scenarios. The speed range is set to [0 50] m/s. We run each simulation instance 20 times with different random seeds, measure the parameters of interest, and report the average results.

We set the transmission power to 7.5 dBm for each UAV. We use the free-space propagation loss model, also known as Friis. The propagation loss of UAV transmission is very well-known to behave close to the Friis model. The constant speed propagation model is also selected as the propagation delay model. UAV nodes use IEEE 802.11b as their wireless communication standard. Table 3 represents the simulation parameters.

We measure the flow success rate, network throughput, and flow completion time to make the comparison. We define flow success rate as the rate of the flows that successfully deliver their entire 5 MB file to the corresponding destination node in the simulation time. It is worth mentioning that if the simulation time is infinite, all flows with any of the routing algorithms can deliver their files successfully. However,

practically it is not feasible to keep the simulation without stop time. Hence, if a flow destination does not receive all of the 5 MB file's packets in the simulation time, we consider that flow as unsuccessful. In the real world, each communication flow may have a deadline. The simulation time is the mapping of the communication deadline concept in the simulation world.

We use the standard definition for throughput, which is the rate of successful packet delivery over the communication bandwidth, and it is measured in Mbps. While we report the average throughput for all successful flows, we find that it might not be fair to report the average of raw throughput values. In some cases, a specific routing algorithm works worse than the others and fails in finding the proper routes. As an instance, it may successfully deliver just one file out of ten. Assume that the source node and the destination be physical neighbors for the successful flow. In this case, it achieves a close to 1 throughput, where the other routing algorithms, which may have a perfect flow success rate, will show much lower throughput. Hence, to report a fair comparison, we weight the throughput with its corresponding flow success rate.

Finally, we define FCT as the time between the first packet of the 5 MB file sent by the source node and the last packet of the same file received by the destination node. Since the simulation time is 500 s, some flows may fail in delivering all of their packets to the corresponding destination in the simulation time. Hence, we consider the FCT equal to 500 for the unsuccessful flows as the lower bound for the FCT of those flows.

4.2. Prediction evaluation

To evaluate the prediction algorithm presented in Section 3.3, we design a simulation scenario and compare its results with those of the extrapolation algorithm as a standard baseline. We use Newton divided differences extrapolation [44] to predict node's position in time $t_2 + \Delta t$, using three consequent node positions in times t_0, t_1, and t_2. It is worthy to note that all other standard extrapolation algorithms, such as Lagrange extrapolation, result in the same value of the Newton divided differences method [45]. We choose this algorithm for its better computation complexity in comparison with the others. Next, we put the predicted position in Eq. (17) to calculate the distance between u_i and u_j at time $t_2 + \Delta t$, and form Eq. (18). Then, we use a lightweight numerical method such as Bisection to find the root and equivalently predict the link lifetime between the corresponding nodes. We measure the prediction error of the link lifetime as the absolute difference between the predicted lifetime and the simulated one.

We consider a network with a different number of nodes where the nodes start moving for a warmup process of 100 s. We perform the prediction and then continue the simulation for extra 500 s. We predict the link lifetimes using both the algorithms of Section 3.3 and the extrapolation. We then watch the links to measure their simulated lifetimes. Next, We calculate the expected value of the error for both methods in each scenario. Obviously, we predict the lifetime for the links that already existed at the time of prediction. We consider the lifetime of the links that last alive until the end of the simulation as the remaining simulation time, i.e. 500 s. We further consider the lifetime of the nodes in which the prediction returned a value more than the simulation time, equal to the remaining simulation time. We run each scenario 1000 times and calculate the expected value of the error and its standard deviation. While the expected value of the error shows the accuracy of the prediction algorithm, the standard deviation shows its precision. The precision represents the interval of the error distribution around the expected value. We find that the curves of all network densities behave almost identical. Hence, we report only the results for a network with 50 nodes.

Fig. 2 shows the cumulative distribution function (CDF) for the results of the expected value and the standard deviation of the error for
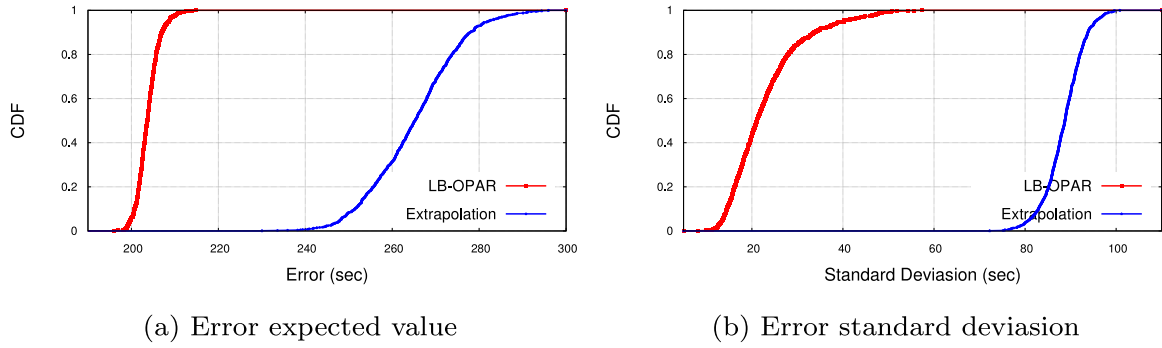
(a) Error expected value

(b) Error standard deviasion

Fig. 2. A comparison of CDF of LB-OPAR prediction algorithm versus extrapolation.

Table 4
Path load weight analysis.

No. of Flows	1	2	3	4	5	6	7	8	9	10
w_3	0	0	0	0.1	0	0.3	0.4	0.7	0.6	0.7
No. of UAVs	55	60	65	70	75	80	85	90	95	100
w_3	0.1	0	0.2	0.1	0.2	0	0	0	0	0.1

Table 5
Weight Analysis for Different Network Loads.

No. of Flows	1	2	3	4	5	6	7	8	9	10
w_3	0	0	0	0.1	0	0.3	0.4	0.7	0.6	0.7
w_1	0.3	0.3	0.3	0.2	0.5	0.4	0.4	0.15	0.2	0.15
w_2	0.7	0.7	0.7	0.7	0.5	0.3	0.2	0.15	0.2	0.15

Table 6
Weight Analysis for Different Network Densities.

No. of UAVs	50	55	60	65	70	75	80	85	90	95	100
w_3	0	0.1	0	0.2	0.1	0.2	0	0	0	0	0.1
w_1	0.5	0.45	0.5	0.4	0.6	0.4	0.5	0.4	0.4	0.5	0.45
w_2	0.5	0.45	0.5	0.4	0.3	0.4	0.5	0.6	0.6	0.5	0.45

both prediction algorithms presented in Section 3.3 and the one use extrapolation. Fig. 2a shows that the prediction algorithm proposed here is on average about 70 s more accurate than the one use extrapolation. Fig. 2b shows the CDF of the standard deviation around the expected value, which represents the prediction precision. The lower standard deviation value shows higher precision, and consequently, a more reliable prediction algorithm. This figure shows that the standard deviation for the proposed prediction algorithm is averagely around 20 s, where this value for the extrapolation algorithm is about 90 s. Regardless of the accuracy of the algorithm, the standard deviation shows that the proposed algorithm is more reliable than that of extrapolation since it has fewer fluctuations.

4.3. Weight analysis

In this Section, we analyze the path length, path lifetime, and path load weights of the LB-OPAR optimization problem, i.e. w_is. Accordingly, we designed a simulation scenario with the same setting of Table 3 where we vary the weights to measure the sensitivity of the network performance versus this variation. In this simulation, we first analyze the path load weight, i.e. w_3. We vary the value of w_3 from 0 to 1 in an increment step of 0.1 and let $w_1 = w_2$. Since $\sum_{i=1}^{3} w_i = 1$, the value of w_1 and w_2 could be calculated easily.

Simulation results show that the highest performance in terms of throughput, success rate, and FCT is achieved for the values of w_3 mentioned in Table 4. The results show that for the network with less load, assigning a value more than zero to w_3 might not benefit the network performance. However, increasing the number of concurrent flows needs a higher weight for w_3 to achieve better performance in terms of all mentioned performance metrics. Furthermore, we find that there is no optimal case with $w_3 = 1$ which means path lifetime and path length always have an obvious effect in choosing the optimal path. We also find that increasing the network density has no significant effect on the value of w_3.

Next, we analyze the effect of varying path-length weight, i.e. w_1, versus path lifetime weight, i.e. w_2, for the fixed path load weight represented in Table 4. By fixing the value of w_3 and varying the values of w_1 and w_2 we find that the best achievable performance is for the values represented in Tables 5 and 6 for different network loads and different network densities, respectively. Table 5 shows that for the network with less number of concurrent communication flows, i.e. less traffic, the path lifetime weight, i.e. w_2 is more important in

comparison with path length. However, results show that the weights of path lifetime and path length are almost equal for the network with higher traffic. Table 5 also shows that the network density has no significant effect on the values of path load and path lifetime weights. Hence, for the network with five concurrent communication flows and different numbers of UAVs, the best performance is achievable when $w_1 = w_2$, in most cases.

4.4. Simulation results

In this section, we compae the LB-OPAR simulation results with those of OPAR, AODV, DSDV, and OLSR, for both RWP and G-M mobility models. We compare the results for flow success rate, network throughput, and flow completion time for different network densities and different network loads, i.e. concurrent flows.

Fig. 3 shows the results for the flow success rate. DSDV and OLSR routing protocols show less than 10% of flow success rate, in most cases, which is mainly because of their inability in catching up with the high dynamicity of the nodes' movement in 3D space. AODV, however, represents better performance in terms of flow success rate in comparison with DSDV and OLSR. It successfully delivers 30% of the flows, in average. Indeed, both OPAR and LB-OPAR significantly outperform the other algorithms by the average of 60% flow success rate. As it is obvious, increasing the network load leads to more significant performance improvement for LB-OPAR compared to OPAR by the average of 5%. For the less number of concurrent flows, since the network load weight is zero or close to zero, no significant difference is shown between LB-OPAR and OPAR, as it is the case for the network with 5 concurrent flows and different densities, i.e. Figs. 3c and 3d. Something worth mentioning is that AODV, DSDV, and OLSR show a higher success rate under the RWP mobility model than G-M. It seems that the randomness of the RWP model helps them to increase their performance. A similar fact was previously discovered in different network studies such as [46]. OPAR and LB-OPAR show no significant
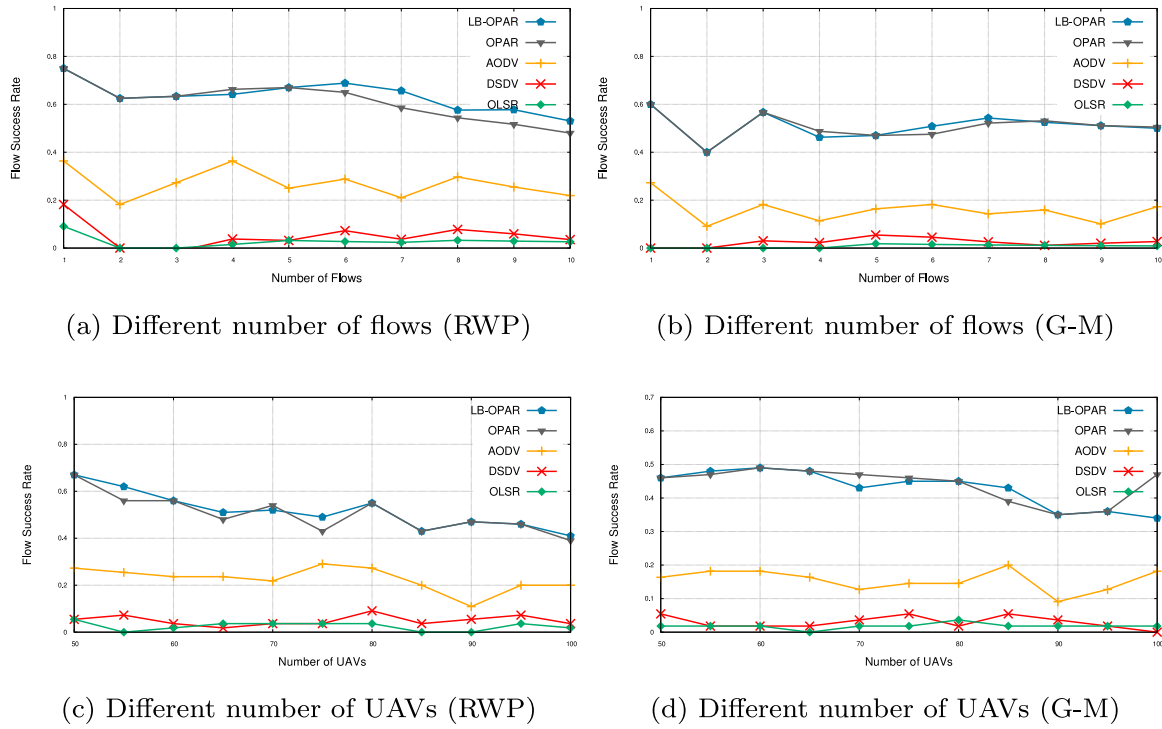
(a) Different number of flows (RWP)

(b) Different number of flows (G-M)

(c) Different number of UAVs (RWP)

(d) Different number of UAVs (G-M)

Fig. 3. A comparison of the flow success rate for different loads and densities.

(a) Different number of flows (RWP)

(b) Different number of flows (G-M)

(c) Different number of UAVs (RWP)

(d) Different number of UAVs (G-M)

Fig. 4. A comparison of the throughput for different loads and densities.

difference under RWP and G-M mobility models, representing their stability of performing well in different conditions.

Next, we investigate the network throughput for all the combinations of different network loads, network densities, and mobility models. Fig. 4 shows the results where the achieved throughput by DSDV and OLSR is close to zero due to their failure in handling the fast topology changes. Interestingly and in contrast to the conventional mobile networks, increasing the number of concurrent flows shows no

significant effect on the throughput, which means in highly dynamic networks, the bottleneck of the throughput is the reroute process due to the topology change not the load of the network. For this parameter, AODV shows much better performance in comparison with DSDV and OLSR, while OPAR and LB-OPAR outperforms AODV by 100% improvements. OPAR and LB-OPAR show competitive behavior. However, LB-OPAR outperforms OPAR in higher network loads and under the RWP mobility model for about 20%. For different network densities,
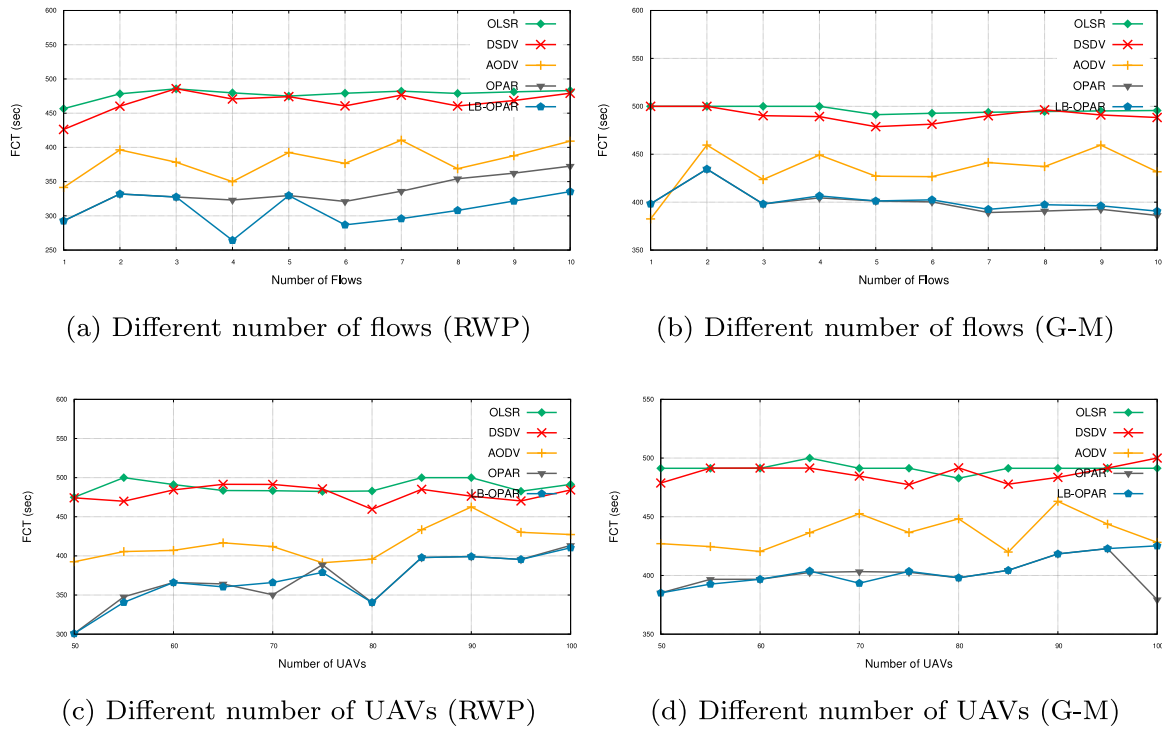
(a) Different number of flows (RWP)

(b) Different number of flows (G-M)

(c) Different number of UAVs (RWP)

(d) Different number of UAVs (G-M)

Fig. 5. A comparison of FCT for different loads and densities.

since there are only five concurrent flows and the network load weight is zero or close to zero, we see no significant differences between LB-OPAR and OPAR. However, they outperform AODV by the average of 100% improvements.

Fig. 5 shows the results for flow completion time. As we mentioned earlier, we consider the simulation time as the lower bound of the FCT for the failed flows. Hence, the results of Fig. 5 show the lower bound for the different algorithms' FCT. This figure includes the combination of all network densities, loads, and the simulated mobility models. For all simulation settings, OPAR and LB-OPAR show the lowest FCT compared to the other routing algorithms. However, LB-OPAR for the network with higher loads shows around 15% less FCT in comparison with OPAR, under RWP mobility model. OPAR and LB-OPAR outperforms AODV by averagely 10%. AODV, in its turn, shows a significantly lower FCT than that of OLSR and DSDV where their average FCT is close to the simulation time, i.e. 500 s, due to their high failure rate. In many cases, OPAR and LB-OPAR show close FCT results, which is one of two cases. The first case is when there are one or a few number of concurrent flows which lead to negligible effect of network load in choosing the optimal path. The latter case owing to the fact that in some cases, there is only one path, or there are several paths, but just one of them has a reasonable objective value, where the others are very long or with a very short lifetime. Hence, both the OPAR and LB-OPAR select the same path, which leads to the close results.

5. Conclusions and future work

The highly dynamic nature of cooperative UAV networks causes the conventional routing algorithms, which aim at finding the shortest path, to fail in reaching the optimal network performance. While we have shown, in our recently published work OPAR, that considering the path lifetime in selecting the routes has a significant impact on the network performance, in this paper, we extend OPAR to consider the network load as well. Accordingly, in this paper, we proposed load-balanced OPAR, an SDN-based routing algorithm for cooperative UAV networks. We modeled the entire problem with an analytical model and proposed a lightweight solution for the proposed model.

We implemented LB-OPAR in the ns-3 network simulator and exhaustively evaluated its performance. We show that LB-OPAR outperforms the benchmark routing algorithms AODV, DSDV, and OLSR. It also improves the performance of OPAR when the network works under higher loads. We further found that the performance bottleneck in highly dynamic UAV networks is basically the route breaks down, more than any other reason. Hence, proposing a routing algorithm to cover the multiple reroutes problem has significant importance in improving cooperative UAV networks' performance. Since LB-OPAR has a lightweight computational and space complexity, we aim at distributing the SDN controller tasks among all network nodes for fully distributed networks as future work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, M. Debbah, A tutorial on UAVs for wireless networks: Applications, challenges, and open problems, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2334–2360.

[2] S. Islam, Q. Huang, F. Afghah, P. Fule, A. Razi, Fire frontline monitoring by enabling UAV-based virtual reality with adaptive imaging rate, in: 2019 53rd Asilomar Conference on Signals, Systems, and Computers, 2019, pp. 368–372, http://dx.doi.org/10.1109/IEEECONF44664.2019.9049048.

[3] H. Peng, A. Razi, F. Afghah, J. Ashdown, A unified framework for joint mobility prediction and object profiling of drones in UAV networks, J. Commun. Netw. 20 (5) (2018) 434–442, http://dx.doi.org/10.1109/JCN.2018.000068.

[4] C. Kerr, R. Jaradat, N.U. Ibne Hossain, Battlefield mapping by an unmanned aerial vehicle swarm: Applied systems engineering processes and architectural considerations from system of systems, IEEE Access 8 (2020) 20892–20903, http://dx.doi.org/10.1109/ACCESS.2020.2968348.

[5] F.Z. Bousbaa, A. Lakas, A.E. Rezigat, H.S. Benguettache, N. Lagraa, C.A. Kerrache, A.E.K. Tahari, GTSS-UC: A game theoretic approach for services' selection in UAV clouds, in: Proceedings of the 2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications, in: DS-RT '21, IEEE Press, 2021, http://dx.doi.org/10.1109/DS-RT52167.2021.9576124.

[6] M.L. Laouira, A. Abdelli, J.B. Othman, H. Kim, An efficient WSN based solution for border surveillance, IEEE Trans. Sustain. Comput. 6 (1) (2021) 54–65, http://dx.doi.org/10.1109/TSUSC.2019.2904855.

[7] T. Dapper e Silva, C.F. Emygdio de Melo, P. Cumino, D. Rosário, E. Cerqueira, E. Pignaton de Freitas, STFANET: SDN-based topology management for flying Ad Hoc network, IEEE Access 7 (2019) 173499–173514, http://dx.doi.org/10.1109/ACCESS.2019.2956724.

[8] H. Peng, Q. Ye, X.S. Shen, SDN-based resource management for autonomous vehicular networks: A multi-access edge computing approach, IEEE Wirel. Commun. 26 (4) (2019) 156–162, http://dx.doi.org/10.1109/MWC.2019.1800371.

[9] E. Bozkaya, B. Canberk, SDN-enabled deployment and path planning of aerial base stations, Comput. Netw. 171 (C) (2020) http://dx.doi.org/10.1016/j.comnet.2020.107125.

[10] S. ur Rahman, G.H. Kim, Y.Z. Cho, A. Khan, Positioning of UAVs for throughput maximization in software-defined disaster area UAV communication networks, J. Commun. Netw. 20 (5) (2018) 452–463, http://dx.doi.org/10.1109/JCN.2018.000070.

[11] Z. Zhao, P. Cumino, A. Souza, D. Rosário, T. Braun, E. Cerqueira, M. Gerla, Software-defined unmanned aerial vehicles networking for video dissemination services, Ad Hoc Netw. 83 (2019) 68–77, http://dx.doi.org/10.1016/j.adhoc.2018.08.023.

[12] G. Secinti, P.B. Darian, B. Canberk, K.R. Chowdhury, Resilient end-to-end connectivity for software defined unmanned aerial vehicular networks, in: 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications, PIMRC, 2017, pp. 1–5, http://dx.doi.org/10.1109/PIMRC.2017.8292772.

[13] A. Ramaprasath, A. Srinivasan, C.-H. Lung, M. St-Hilaire, Intelligent wireless ad hoc routing protocol and controller for uav networks, in: Y. Zhou, T. Kunz (Eds.), Ad Hoc Networks, Springer International Publishing, Cham, 2017, pp. 92–104.

[14] C. Perkins, E. Belding-Royer, S. Das, Ad hoc On-Demand Distance Vector (AODV) Routing, RFC 3561, Network Working Group, 2003, pp. 1–37, URL https://tools.ietf.org/html/rfc3561.

[15] C.E. Perkins, P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications, 24 (4), 1994, pp. 234–244, http://dx.doi.org/10.1145/190809.190336.

[16] T. Clausen, P. Jacquet, Optimized Link State Routing Protocol (OLSR), RFC 3626, Network Working Group, 2003, pp. 1–75, URL https://tools.ietf.org/html/rfc3626.

[17] M.F. Nowlan, N. Tiwari, J. Iyengar, S.O. Aminy, B. Fordy, Fitting square pegs through round pipes: Unordered delivery wire-compatible with TCP and TLS, in: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, in: NSDI'12, USENIX Association, USA, 2012, p. 28.

[18] M. Gharib, F. Afghah, E. Bentley, OPAR: Optimized predictive and adaptive routing for cooperative uav networks, in: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, 2021, pp. 1–6, http://dx.doi.org/10.1109/INFOCOMWKSHPS51825.2021.9484489.

[19] 2020. Link. URL https://www.nsnam.org/.

[20] A. Boukerche, B. Turgut, N. Aydin, M.Z. Ahmad, L. Bölöni, D. Turgut, Routing protocols in ad hoc networks: A survey, Comput. Netw. 55 (13) (2011) 3032–3080, http://dx.doi.org/10.1016/j.comnet.2011.05.010.

[21] J.M.M. Biomo, T. Kunz, M. St-Hilaire, Routing in unmanned aerial ad hoc networks: Introducing a route reliability criterion, in: IFIP Wireless and Mobile Networking Conference, 2014, pp. 1–7.

[22] X. Li, J. Yan, LEPR: Link stability estimation-based preemptive routing protocol for flying ad hoc networks, in: 2017 IEEE Symposium on Computers and Communications, ISCC, 2017, pp. 1079–1084.

[23] D.S.J. De Couto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, in: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, in: MobiCom '03, Association for Computing Machinery, New York, NY, USA, 2003, pp. 134–146, http://dx.doi.org/10.1145/938985.939000.

[24] C. Pu, Link-quality and traffic-load aware routing for UAV ad hoc networks, in: 2018 IEEE 4th International Conference on Collaboration and Internet Computing, CIC, 2018, pp. 71–79.

[25] M. Song, J. Liu, S. Yang, A mobility prediction and delay prediction routing protocol for uav networks, in: 2018 10th International Conference on Wireless Communications and Signal Processing, WCSP, 2018, pp. 1–6.

[26] S. Rosati, K. Krużelecki, G. Heitz, D. Floreano, B. Rimoldi, Dynamic routing for flying ad hoc networks, IEEE Trans. Veh. Technol. 65 (3) (2016) 1690–1700.

[27] N. Lyu, G. Song, B. Yang, Y. Cheng, QNGPSR: A Q-network enhanced geographic ad-hoc routing protocol based on GPSR, in: 2018 IEEE 88th Vehicular Technology Conference, VTC-Fall, 2018, pp. 1–6, http://dx.doi.org/10.1109/VTCFall.2018.8690651.

[28] B. Karp, H.T. Kung, GPSR: Greedy perimeter stateless routing for wireless networks, in: MobiCom '00, Association for Computing Machinery, New York, NY, USA, 2000, pp. 243–254, http://dx.doi.org/10.1145/345910.345953.

[29] J. Liu, Q. Wang, C. He, K. Jaffrès-Runser, Y. Xu, Z. Li, Y. Xu, QMR: Q-learning based multi-objective optimization routing protocol for flying ad hoc networks, Comput. Commun. 150 (2020) 304–316, http://dx.doi.org/10.1016/j.comcom.2019.11.011.

[30] J. Liu, Q. Wang, C. He, Y. Xu, ARdeep: Adaptive and reliable routing protocol for mobile robotic networks with deep reinforcement learning, in: 2020 IEEE 45th Conference on Local Computer Networks, LCN, 2020, pp. 465–468, http://dx.doi.org/10.1109/LCN48667.2020.9314848.

[31] A review of AI-enabled routing protocols for UAV networks: Trends, challenges, and future outlook, Ad Hoc Netw. 130 (2022) 102790, http://dx.doi.org/10.1016/j.adhoc.2022.102790.

[32] K. Sundar, S. Rathinam, Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots, IEEE Trans. Autom. Sci. Eng. 11 (1) (2014) 287–294.

[33] A. Coelho, E.N. Almeida, P. Silva, J. Ruela, R. Campos, M. Ricardo, RedeFINE: Centralized routing for high-capacity multi-hop flying networks, in: International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob, 2018, pp. 75–82.

[34] B.N. Coelho, V.N. Coelho, I.M. Coelho, L.S. Ochi, R.H. K., D. Zuidema, M.S. Lima, A.R. da Costa, A multi-objective green UAV routing problem, Comput. Oper. Res. 88 (2017) 306–315.

[35] Y. Cheriguene, S. Djellikh, F.Z. Bousbaa, N. Lagraa, A. Lakas, C.A. Kerrache, A.E.K. Tahari, SEMRP: An energy-efficient multicast routing protocol for UAV swarms, in: Proceedings of the IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications, in: DS-RT '20, IEEE Press, 2020, pp. 182–189.

[36] G. Secinti, P.B. Darian, B. Canberk, K.R. Chowdhury, SDNs in the sky: Robust end-to-end connectivity for aerial vehicular networks, IEEE Commun. Mag. 56 (1) (2018) 16–21, http://dx.doi.org/10.1109/MCOM.2017.1700456.

[37] C. Singhal, K. Rahul, Efficient QoS provisioning using SDN for end-to-end data delivery in UAV assisted network, in: 2019 IEEE International Conference on Advanced Networks and Telecommunications Systems, ANTS, 2019, pp. 1–6, http://dx.doi.org/10.1109/ANTS47819.2019.9118086.

[38] X. Zhang, H. Wang, H. Zhao, An SDN framework for UAV backbone network towards knowledge centric networking, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, 2018, pp. 456–461, http://dx.doi.org/10.1109/INFCOMW.2018.8406959.

[39] W. Qi, Q. Song, X. Kong, L. Guo, A traffic-differentiated routing algorithm in flying ad hoc sensor networks with SDN cluster controllers, J. Franklin Inst. B 356 (2) (2019) 766–790, http://dx.doi.org/10.1016/j.jfranklin.2017.11.012, Special Issue on Modeling, Analysis and Control of Networked Autonomous Agents.

[40] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, 1972, pp. 85–103.

[41] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., The MIT Press, 2001.

[42] J.J. Durillo, H.M. Fard, R. Prodan, MOHEFT: A multi-objective list-based method for workflow scheduling, in: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, 2012, pp. 185–192, http://dx.doi.org/10.1109/CloudCom.2012.6427573.

[43] M.R. Bouzidi, M. Daoudi, B. Ziani, K. Boukhalfa, C.A. Kerrache, N. Lagraa, FAMOBACH: A fast and survivable workflow scheduling approach based MOHEFT using backtacking and checkpointing, Comput. Commun. 171 (2021) 16–27, http://dx.doi.org/10.1016/j.comcom.2021.02.005.

[44] E.E. Kalu, Numerical Methods with Applications: Abridged, 2010.

[45] J. Kiusalaas, Numerical Methods in Engineering with Python, Cambridge University Press, 2005.

[46] A. Singla, C.-Y. Hong, L. Popa, P.B. Godfrey, Jellyfish: Networking data centers randomly, in: 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 12, USENIX Association, San Jose, CA, 2012, pp. 225–238.

Mohammed Gharib is a research scholar at the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, US. He received the B.S. degree from the Department of Computer Engineering, Baghdad University of Technology, Baghdad, Iraq, in 2007. He received the M.S. and Ph.D. degrees from the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, in 2009 and 2015, respectively. He was also a postdoctoral researcher in the School of Computer Science, Institute for Research in Fundamental Science (IPM), Tehran, Iran, between 2016 to 2019; a postdoctoral researcher in the Department of Computer Science, Johns Hopkins University, Batimore, MD, US, 2019–2020; and a postdoctoral researcher in the School of Informatics, Computation, and Cyber Systems, Northern Arizona University, Flagstaff, AZ, US, 2020–2021. His research interests include multi-hop wireless networks, name data networking, cloud network, and their security aspects.

Fatemeh Afghah is an Associate Professor with the Department of Electrical and Computer Engineering, Clemson University. She was affiliated with the School of Informatics, Computing and Cyber Systems, Northern Arizona University (NAU), Flagstaff, AZ, USA, where she is the Director of Wireless Networking and Information Processing (WiNIP) Laboratory. Her research interests include wireless communication networks, decision making in multi-agent systems, radio spectrum management, PUF-based security, and artificial intelligence in healthcare. Her research has been continually supported by NSF, AFRL, AFOSR, NIH and ABOR. She is the recipient of several awards including the Air Force Office of Scientific Research Young Investiga-

tor Award in 2019, NSF CAREER Award in 2020, NAU's Most Promising New Scholar Award in 2020, and NSF CRII Award in 2017. She is the author/co-author of over 90 peer-reviewed publications and served as the associate editor for several journal including Elsevier Ad hoc networks, Springer Neural Processing Letters and the organizer and TPC chair for several international IEEE workshops in the field of UAV communications and AI, including IEEE INFOCOM Workshop on Wireless Sensor, Robot, and UAV Networks (WiSRAN'19), IEEE WOWMOM Workshop on Wireless Networking, Planning, and Computing for UAV Swarms (SwarmNet'20&21), and 2021 NSF Smart Health PI workshop on "Smart Health in the AI and COVID Era".

Elizabeth Serena Bentley has a B.S. degree in Electrical Engineering from Cornell University, a M.S. degree in Electrical Engineering from Lehigh University, and a Ph.D. degree in Electrical Engineering from University at Buffalo. She was a National Research Council Post-Doctoral Research Associate at the Air Force Research Laboratory (AFRL) in Rome, NY. Currently, she is employed by the AFRL Information Directorate, performing in-house research and development in the Communication Technology & Systems Branch and managing the Cross-Layer Heterogeneous Autonomous Resilient On-Demand Networks (CHARON) program that focuses on mission responsive swarm networking. Her research interests are in cross-layer optimization, swarm networking, directional networking, wireless multiple-access communications, and modeling and simulation.