Dynamic Controller Placement in Software Defined Drone Networks

Mohannad Alharthi
School of Computing
Queen's University
Kingston, Ontario
harthi@cs.queensu.ca

Abd-Elhamid M. Taha
Electrical Engineering Department
Alfaisal University
Riyadh, Saudi Arabia
ataha@alfaisal.edu

Hossam S. Hassanein
School of Computing
Queen's University
Kingston, Ontario
hossam@cs.queensu.ca

***Abstract*—Controller placement is one of the most important aspects in Software Defined Networking (SDN) as it is critical in determining the performance of the network. While SDN is mainly applied in wired networks, we attempt to utilize it in a futuristic drone-based network architecture to capitalize on the programmability and flexibility offered by SDN. Such drone network comprises a multi-hop network of drones as the nodes of the network which also act as programmable network nodes and forwarding elements. In such a dynamic and mobile network, maintaining SDN controller connectivity becomes a challenge, especially when the drone network requires operating independently from a ground infrastructure for some deployment scenarios. In this work, we attempt to address this challenge by implementing a dynamic scheme for controller placement that deploys a minimum number of drones that operate as SDN controllers and adjust their locations dynamically as the controlled nodes adjust their locations to meet changing mission requirements.**

I. Introduction

The introduction of Software Defined Networking (SDN) enabled powerful adaptive network programmability as well as flexible network management. SDN enables this power by separating the control and data plane by turning network devices (switching devices) into simple but flexible devices programmable via a standard API (such as OpenFlow). The control plane, comprised of logically centralized controllers, implement the control logic and dynamically program the data plane, effectively able to implement a wide range of adaptive functionalities. The decoupling of the control and the data planes allows for central network management that is unified and flexible regardless of the underlying hardware.

SDN is mainly utilized in data centers and wide area networks (WANs) [1]. However, SDN has been proposed to be utilized in other networking paradigms such as in wireless networks [2] [3]. In our work, we proposed an architecture for software defined drone networks [4]. This architecture enables flexible and reusable infrastructure for various deployments of drone networks such as for civilian monitoring, and as a flying network infrastructure in remote areas and for temporary or emergency situations. The architecture is based on SDN to take advantage of programmability and flexibility in management and configuration, and the ability to implement adaptive network solutions according to current state and environment, accommodating the dynamic nature of drone networks. We briefly review this architecture in Section III-A.

The SDN controller is a key component in the network as it has a direct effect on network performance, especially in a dynamic environment. In such a dynamic network, maintaining SDN controller connectivity becomes a challenge. As the network has some degree of mobility, it has to maintain a connection to controllers located in the ground infrastructure, and this can limit the where nodes can be positioned. This is especially challenging when the drone network requires operating independently from the ground infrastructure for some deployment scenarios, e.g., in a remote area or where no other infrastructure is available. The deployment of additional drones as SDN controllers gives flexibility in network deployment locations and movements away from a ground control station, as this also ensures a reliable connection to the controller with minimum delay.

In this work, we attempt to address this challenge by implementing a dynamic scheme for controller placement that deploys a minimum number of drones that operate as SDN controllers, and adjust their locations dynamically as the controlled nodes adjust their locations during the mission. This may impose additional costs in terms of dedicating a number of controllers for SDN control while ensuring their processing capacity and availability to network nodes. The dynamic location changes also consume drones power resources so movement needs to be minimized.

The contribution of this work is twofold. First, we provide a controller placement scheme that initially places a minimum number of drone SDN controllers in the space of the deployment area within the budget of network operators, and assigns each node to a controller within a threshold of link quality metric, while ensuring that controllers are not over capacitated. The placement prioritizes one-hop inter-controller links but it also flexible to allow for overlay links. Second, as locations of network nodes change, we update the assignment while maintaining original placement constraints, and when necessary we update controller locations to make them available to moved nodes, while minimizing the total distances traveled by controllers to make them available to nodes as soon as possible without requiring high speeds to potentially minimize energy consumption.

978-1-7281-0962-6/19/$31.00 ©2019 IEEE

II. Related Work

SDN controller placement is one of the well-studied topics in the SDN literature in datacenter and WAN contexts. The work by Heller et al. [5] introduced the controller placement problem (CPP) considering average latency between switches and controllers, then others followed suit. Existing works differ in the placement metrics used, such as switch-controller latency [5], controller capacity [6], and deployment cost [7]. Other works focused more on the reliability, fault tolerance, and availability of controllers [8][9]. CPP is also studied considering inter-controller connectivity and the resilience of the distributed control plane. One of the most recent works in this aspect is [9], which targets Software-Defined WANs (SD-WANs). In [9], Tanha et al. presented a capacity-aware placement scheme that places and assigns multiple controllers to each switch while guaranteeing switch-controller and inter-controller delay. An important aspect in CPP is the dynamic adjustment to controller placement and assignment to switches considering current conditions of the network. In [7], authors build a dynamic controller provisioning scheme that runs at specified periods, activating and deactivating controllers depending on current demand. The placement in this scheme considers various costs in the network including flow-setup time and statistics collection, as well as interactions between controllers in terms of state synchronization.

In [10], Abdel-Rahman et al. introduced the wireless CPP (WCPP) where links between switches and controllers are wireless. A stochastic approach was used to model the uncertainties of the wireless channel and the controller processing delay, with the goal of restricting per-link response time and minimizing the number of controllers. In [11], Dvir et al. introduced another model for WCPP using a new placement metric called *transparency*, which is the marginal average latency in the data plane caused by the interference from controllers. The objective of their model is minimizing the average outage of control links and the average latency while constraining the link throughput and transparency below certain thresholds.

In this work, we apply SDN in a different context, which is a drone-based network. While incorporating SDN into this domain has been discussed in works such as in [12], the deployment of a control plane is not studied. As discussed in III-A, certain deployments may require deploying drones as controllers to increase the flexibility of the network. This introduces additional challenges as such deployments require considering the cost of deployment and the mobility of the SDN network. Controller placement needs to be dynamically adjusted so that all controlled nodes have reliable access to controllers. None of the above works can be applied directly to our architecture. In our architecture, the controllers are placed anywhere in space, as opposed to existing works, in which controllers are assumed to be located within any node in evaluated topologies. As well, the mobility of network nodes needs to be taken into account.

III. System Overview

In this section, we describe the framework and architecture of the SDN-based drone network. We also describe the deployment scenario for the network and the system operations that take place for deploying and placing SDN controllers. Then, we define the placement problem given the framework and architecture described.

A. Network Architecture

The architecture depicted in Fig. 1 provides a reconfigurable and reusable network for different drone applications. Drones comprise the nodes of a multi-hop wireless network. Drones are designated as either network drones or task drones. Network drones act as switches in a typical SDN network forwarding traffic. Network drones are SDN-enabled and programmable via an API such as OpenFlow. SDN is utilized to enable programmability and configuration of network nodes and to manage the forwarding of traffic using multiple network interfaces installed on drones. Task drones perform the actual tasks related by the mission, such as sensing, monitoring, or providing wireless access to ground users. The network is managed by a ground management system that carries out the planning and deployment of drones, and the monitoring, management, and control of the network during the mission. The management system hosts the main SDN controller of the network. The southbound interface (OpenFlow) is extended to allow for configuring various aspects of drones such as flight control in addition to networking. These aspects are implemented as SDN applications that implement and configure topology and flight control, networking, and configuring task-specific functions. The network is sufficiently flexible to enable deploying the network beyond the reach of the ground system. In such cases, SDN controllers and applications are deployed along with the network on controller drones as discussed in III-B. We refer to network drones (all except controllers) as nodes and drones interchangeably.

B. Deployment Scenario

We consider a specific deployment scenario of the drone-based network. The drone network is deployed to provide communication and connectivity at a remote area where all or most of cellular networking infrastructure is not available. Examples of such situations include missions targetting natural disasters and remote scientific missions where some form of a communication network is needed. In such settings, network drones have a limited degree of mobility, such that they hover most of the time to maintain a certain topology, and then their locations are adjusted at some points of time to enhance coverage or to accommodate new requirements. The deployed network requires the deployment of drones as SDN controllers. This is due to the impracticality of using a fixed SDN controller on the ground due to lack of network infrastructure. Additionally, the mission location may lack the proximity needed to ground satiations to communicate with SDN controllers.

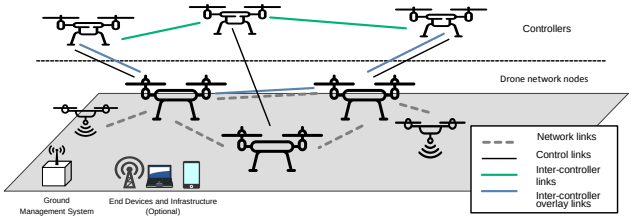

Fig. 1. The architecture of the network and overview of controller placement

C. Problem Definition

A controller deployment scheme is needed to determine the number and locations of controller drones and their assignment to nodes. During the operation of the network, the formation of the drone network may change, and controller links may break as a result of changes. Then, drone-controller assignment needs to be updated. Only updating assignment using the same controller locations may not be feasible (not all moved drones can establish links to controllers). In such cases, controller locations need to be recomputed to accommodate changes in the drone network and to allow establishing links with their assigned drones. Since controllers are already deployed, it will need to travel to their new locations in the shortest possible time to become available to their assigned drones. Controller travel distance needs to be minimized, as traveling long distances at high speeds can drain energy quickly.

We assume without loss of generality that the topology formation for the drone network is computed independently from the deployment scheme for SDN controllers. Network operators are free to use a topology formation scheme for network drones without considering controller deployment.

IV. System Model

The drone network comprises a connected and undirected graph $G = (D, E)$, where the vertices set $D = \{d_1, d_2, \ldots, d_{|D|}\}$ is the set of network drones which are also considered SDN switches, and E is the set of edges representing the wireless links formed between each pair $d_i, d_j \in D$. $|D|$ is the number of drones in the network. Each network drone d_i generates a number of requests to its controller. The rate of requests in a unit of time is d_i^{rate}. All controller drones have a capacity for processing drone requests expressed as $S_{capacity}$.

The network system will deploy a number of controllers limited by the maximum number of controllers to deploy N, which is defined by the network operators according to available budget. The resulting number of controller drones that will actually be deployed is N_d. Controllers can be deployed anywhere in the 2D plane of the coverage area at a suitable altitude. They attach to the network graph G through the nearest drone d_i. L is the set of all allowable controller locations, where $l_i \in L$ represents a possible 2-D coordinates for a controller location. The placement will result in $C = \{c_1, c_2, ...c_{N_d}\}$, which defines the resulting set of controllers deployed. Each $c_i \in C$ is mapped with its placement location $l_i \in L$.

The wireless communication model between all drones including controllers is based on a Friis-based free space propagation model, where path loss, in dB, over transmission distance d is defined as [13]:

$$Pl(d) = Pl(d_0) + nlog_{10}(\frac{d}{d_0}) \quad (1)$$

where d_0 is the reference distance (1m), and n is the path loss exponent ($n = 3$), and $P_l(d_0)$ is the Friis transmission equation defined as:

$$Pl(d_0) = \frac{P_t G_t G_r \lambda^2}{16\pi^2 d_0^2 L} \quad (2)$$

where P_t is the transmission power, G_t and G_r are the transmission and reception gains (1 dBm), λ is the wave length, and L is the system loss (equals 1).

A. Initial Controller Placement and Assignment

Initially, the ground station deploys the drone network and determines its topology and physical locations for drones. Once a need for a flying control plane is determined, a placement will be computed for controller drones. The number of controller drones and their locations will be determined, as well as the assignment of each node to a controller. The joint placement and assignment scheme deploys a minimum number of controllers, limited by the number of available drones that can function as controllers. The assignment of drones to controllers is based on link quality between them.

For connectivity between controllers, each controller must maintain a connection to every other controller to enable synchronization of the state of the network. Ideally, direct connections between controllers are best to reduce latency. However, this can make the placement difficult to achieve or infeasible given a limited number of controller and how drones are distributed in space. A possible approach is that controllers utilize multi-hop connections through the actual network, forming overlay links on underlying drone network links when it is difficult to form direct connections. However, this approach may overuse much-needed network and computational resources. A balanced approach can be placing controllers in a way that maximizes single-hop connections and leverage multi-hop connections to form a complete graph for inter-controller links as shown in Fig. 1.

B. Placement Problem Formulation

Here we describe the formulation for the initial placement problem, which is a joint controller placement and assignment problem. Every placement of a controller at l_i and its assignment to any d_i is constrained by achieving a maximum value for a link quality metric for the wireless control channel that need to be formed between them. Q_{d_i,l_j} represents the quality of wireless channel that can be achieved between d_i and a controller placed at l_j. The quality metric can be the transmission delay or channel path loss. The threshold for this required measurement is given by Q_m.

We model inter-controller links by considering the cost of direct and indirect links between any pair at l_i, l_j. A direct

link can be established if $Q_{l_i,l_j} \leq Q_m$. If a direct link cannot be achieved, an indirect (overlay) link can exist if a multi-hop link exists between drones that controllers at l_i, l_j can connect to (separate from control links). We express such cost of any inter-controller links by $L_{cc}^{l_i l_j}$, which corresponds to $hops_{l_i l_j}$, the number of hops between controllers at l_i, l_j:

$$L_{cc}^{l_i l_j} = \begin{cases} 1 & \text{if } Q_{l_i l_j} \leq Q_m \\ hops_{l_i l_j} & \text{if } hops_{l_i l_j} \geq 1 \text{ and } Q_{l_i l_j} > Q_m \end{cases} \quad (3)$$

We define $y_{l_i} = \{0, 1\}$ as a binary variable to indicate if a controller is placed at l_i. We also define the binary variable $x_{d_i l_j} = \{0, 1\}$ to express assignment of drone d_i to a controller located at l_j. We also define the binary variable $a_{d_i} = \{0, 1\}$ to indicate whether d_i is assigned to any controller.

Then, the objective function aims to minimize the number of deployed controllers:

$$N_d = \sum_{l_j \in L} y_{l_j} \quad (4)$$

We attempt to minimize the sum of all $Q_{d_i l_j}$ associated with assignments Pl_{total}, which we found it improves assignments by making assigned d_i, l_j pairs closer:

$$Pl_{total} = \sum_{d_i \in D} x_{d_i l_j} \cdot Q_{d_i l_j} \quad (5)$$

We also minimize the number of hops for inter-controller connections, which increases chances one-hop connections ($L_{cc}^{l_i l_j} = 1$):

$$H_{total} = \sum_{l_i, l_j \in L, i \neq j} y_{l_i} y_{l_j} L_{cc}^{l_i l_j} \quad (6)$$

Then, the model becomes (program CPlace):

$$\min_{x,y} \quad w_1 N_d + w_2 Pl_{total} + w_3 H_{total} \quad (7)$$

subject to

$$x_{d_i l_j} \leq y_{l_j} \quad (8)$$

$$a_{d_i} = \text{or}\{x_{d_i l_{j=1}}, \ldots, x_{d_i l_{j=|L|}}\}, \forall d_i \in D \quad (9)$$

$$x_{d_i l_j} \cdot Q_{d_i l_j} \leq Q_m \quad (10)$$

$$\sum_{l_j \in L} x_{d_i l_j} \leq Z_{max}, \forall d_i \in D \quad (11)$$

$$\sum_{d_i \in D} x_{d_i l_j} \cdot d_i^{rate} \leq S_{capacity}, \forall l_j \in L \quad (12)$$

$$\sum_{l_j \in L} y_{l_j} \leq N \quad (13)$$

$$y_{l_j} y_{l_j} L_{cc}^{l_i l_j} \leq hops_{max}, \forall l_i, l_j \in L, i \neq j \quad (14)$$

$$\sum_{d_i \in D} a_{d_i} = |D| \quad (15)$$

where w_1, w_2, w_3 are weight factors used to adjust the terms of the objective function.

The first constraint (8) expresses the assignment and placement relationship where d_i can be assigned to a controller at l_j only if a controller is placed at l_i. The constraint in (9) ensures that $a_{d_i} = \{0, 1\}$, which is derived from all $x_{d_i l_j}$ values using logical OR operator. The equivalent linear programming constraints for this expression is omitted for simplicity. Using the constraint in (10), the drone-controller assignment is limited by the required path-loss Q_m for the link between them. The constraint in (11) ensures that each drone can be assigned to a predefined number of controllers Z_{max}. In this work, $Z_{max} = 1$. Then, using the constraint in (12), each controller drone is assigned a number of drones with demand d_i^{rate} below its processing capacity. The constraint in (13) limits the number of placed controllers to the number of available controllers N, and the constraint in (14) selects inter-controller links. Finally, we ensure that all drones are assigned to controllers using the constraint in (15).

C. Dynamic Reassignment and Placement Scheme

The dynamic scheme first aims to compute updated assignments for drones in new locations using already deployed controllers without changing controller locations. This is accomplished using a reassignment algorithm that abides by the constraints of the initial placement. For drones that cannot be assigned to controllers at existing locations, we attempt to change controller locations with minimum movements to preserve their energy and minimizing the time needed for controllers to reach their new locations. The algorithmic framework for the dynamic scheme is shown in Algorithm 1.

The Reassign algorithm described in Algorithm 2 attempts to assign drones to existing controllers without moving them. The algorithm receives a set of updates U, each u_i is expressed as a tuple $\langle \bar{d}_i, loc_{d_i} \rangle$, where $\bar{d}_i$ is the drone that will have its position changed to loc_{d_i}. The algorithm iterates the list of updates, and for each updated drone, first it checks whether the currently assigned controller is still able to communicate with the drone at the new location. If it is, then the algorithm continues to the next u_i (lines 3-6). If the drone is not able to communicate with its assigned controller at the new location, then the algorithm, utilizing its global view of the other controller locations, lists all controllers and computes their achievable $Q_{l_i d_i}$. If it satisfies Q_m, then it is added to a list of controllers that it is possible to be assigned to. The list is sorted in ascending order, then we traverse the list and pick the first controller that can also satisfy the capacity constraint (lines 7-20).

If there are remaining nodes that cannot be assigned to any current controllers, then we recompute a new placement and assignment for controllers. To achieve this, we modify the initial placement model described in IV-B. We limit N by the number of deployed controllers N_d in constraint (13) as we

Algorithm 1 DynAssignPlace

1: $Reassign(U)$
2: **if** there are unassigned nodes **then**
3: $\quad sol = CReloc()$
4: **else if** sol is infeasible **then**
5: $\quad CPlace()$
6: **end if**

are only repositioning existing controllers. We remove the first term of the objective function and add the term:

$$Dist_{total} = \sum_{l_j \in L} y_{l_j} \cdot dist(l_j, \bar{l_j}) \quad (16)$$

where $dist(a, b)$ is the distance between a and b. We map each candidate location $l_j \in L$ to the closest location of an existing controller (from locations obtained in the previous solution). This effectively clusters L into k clusters ($k = N_d$). Each of the resulting clusters $S = \{S_1, S_2, \ldots, S_k\}$, where each $S_i \subset L$, is assigned to an existing controller location. We limit each cluster to one controller placement, and consider each placement a new location for each existing controller. This allows us to map new placements to existing ones and compute the distance difference, and then minimize how far a controller can move. We achieve this using the constraint:

$$\sum_{l_j \in S_i} y_{l_j} = 1, \forall S_i \in S \quad (17)$$

Then, we have the following (program CReloc):

$$\min_{x,y} \quad w_2 Pl_{total} + w_3 H_{total} + w_4 Dist_{total} \quad (18)$$

$$\text{subject to} \quad (8)\text{-}(15), (17) \quad (19)$$

If no feasible solution can be obtained, then we resort to applying the initial placement program with the original objective function, without limiting N by N_d and without constraint (17). Then we consider any extra controllers as new ones that need to be deployed.

V. Performance Evaluation

In this section, we describe our evaluation process and results. The optimization model and reassignment algorithm were implemented using Python and Gurobi solver (v8.1.1). The optimization model contains quadratic terms in the objective function and quadratic constraints. Gurobi can solve models with quadratic objective terms and constraints. Our model gets solved in 3 to 19 seconds approximately depending on $|D|$ and $|L|$. Evaluations were ran on a laptop equipped with a dual-core 2.4 GHz Core i5 CPU and 16 GB of RAM.

Algorithm 2 Reassign

```
Output: new drone assignments for U, unassigned drones
unassignedDrones ← empty
for u_i ∈ U do
    if path loss for current assignment ≤ Q_m then
        continue
    end if
    nearbyCtls ← empty
    for j = 0 to N_d do
        if pathloss for c_j to drone new loc ≤ Q_m then
            add c_j to nearbyControllers
        end if
    end for
    sort nearbyCtls by path loss to new drone location
    while nearbyCtls not empty do
        ctl ← pop(nearbyCtls)
        if ctl.load + d̄_i^rate ≤ S_capacity then
            assign d̄_i to ctl, continue
        end if
    end while
    add to unassigned drone list
end for
```

TABLE I
Evaluation Parameters

Parameter	Value	Parameter	Value
P_t	25 dBm	Frequency	2 GHz
$\|D\|$	30, 60, 90 nodes	Area	1×1 km^2
N	10	Controller cell size	100×100 m^2
Q_m	85, 90, 95 dB	$S_{capacity}$	5000 req/s
d_i^{rate}	100 req/s	w_1, w_2, w_3, w_4	100, 1, 50, 1

To simplify solving the model, we discretize controller placement locations. The area is divided into a grid of $w \times h$ cells. Each $l_i \in L$ is positioned in the center of its cell. We also remove any locations that do not satisfy Q_m. We assume controllers are at a higher altitude than regular nodes. We also precompute the values for $L_{cc}^{l_i,l_j}$ before each scenario by computing $hops_{l_i l_j}$ through nearest $d \in D$ and shortest paths in G.

First, we evaluated initial placements. We ran three experiments with three topologies of different sizes $|D|$ shown in Table I. Each topology was tested in three scenarios associated with three values of Q_m. We used randomly generated graphs for the drone network, where nodes are placed in the simulation area in a uniform distribution. Links between nodes were formed between every pair that can achieve a path loss below 100 dB, and limited by a maximum node degree of 4. Only connected graphs were selected. All simulation parameters are shown in Table I. Fig. 2 shows the results of the initial placements in terms of the number of deployed controller drones, the average path loss for the channel between controllers and assigned nodes, the average controller load, and the number of hops for inter-controller links.

Second, we evaluated the dynamic scheme. We used a single randomly generated topology for the drone network where $|D| = 40$. We ran the CPlace program to initially place controllers using parameters $Q_m = 90$ dB and controller cell size of 50×50 m^2. The rest of the parameters are the same as shown in Table I.

After the initial controller placement, we applied ten successive topology update events to simulate changes in network node locations. In each event, a random number of drones is selected and moved to a random point within a 300×300 m^2 around each drone, as it is not intended for drones to move anywhere across the network area. Drones are expected to move only within a certain range to improve coverage or communication. According to our system operation, the topology control component sends the list of drones to be moved, U, to a controller to trigger the dynamic reassignment scheme (Algorithm 1).

We compared the dynamic scheme with two other baseline schemes. Here, we refer to our purposed dynamic scheme as

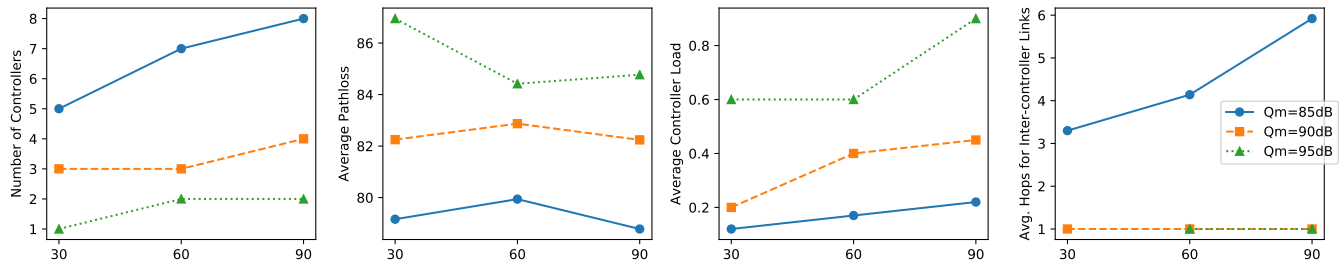

Fig. 2. Outcomes of initial placements for different topologies with $|D| = 30, 60, 90$ nodes and Q_m values

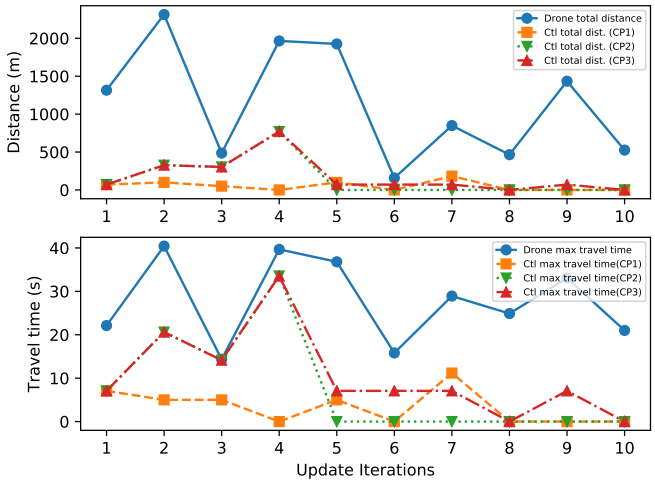

Fig. 3. Result of the dynamic placement scheme

CP1, while the baseline schemes are CP2 and CP3. CP2 is similar to CP1 but CP2 only executes Algorithm 2, then if not all drones could be assigned, it applies CPlace (7), i.e., it does not optimize for controller relocation distance. CP3 only applies program CPlace (7) with each topology update event (The initial placement program is reapplied with each event.)

The initial placement resulted in three controllers, with single-hop inter-controller connections, and average controller load of 0.26, and average path loss 82 dB.

The results of the dynamic scheme are shown in Fig. 3. The top graph shows the total distances travelled by controllers after each update event for all schemes against the total distances travelled by network nodes. The bottom graph shows the time it takes for the update event to take place for the node traveling the longest distance and the time it takes for controller drones to relocate in response. We assume a speed of 10 m/s for all drones including controllers. The total distances travelled by controllers for are 503.22 for CP1, 1471.44 for CP2, and 1754.28 for CP3 in meters. This shows a significant reduction of relocation distance compared to successive iterations of placements, which is attributed to optimizing placement for relocation distance. Through our tests, the dynamic scheme rarely requires falling back to CPlace.

VI. Conclusion

In this paper, we introduced a controller placement scheme for drone-based SDN networks. The scheme initially places a minimum number of controllers while optimizing path loss for node-controller links and maximizing one-hop links between controllers. The placement is dynamically adjusted to accommodate underlying network topology changes. Our evaluation shows that the dynamic scheme minimizes the relocation distance of controllers, leading to minimizing the time of relocation and potentially preserving energy of controllers.

Acknowledgment

The second author would like to acknowledge the financial support of Alfaisal University through its Scientific Conference Attendance Award.

References

[1] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," *SIGCOMM Comput. Commun. Rev.*, vol. 43, pp. 3–14, aug 2013.

[2] H. Ghafoor and I. Koo, "CR-SDVN: A cognitive routing protocol for software-defined vehicular networks," *IEEE Sensors J.*, vol. 18, no. 4, pp. 1761–1772, Feb 2018.

[3] C. J. Bernardos *et al.*, "An architecture for software defined wireless networking," *IEEE Wireless Commun.*, vol. 21, no. 3, pp. 52–61, June 2014.

[4] M. Alharthi, A. M. Taha, and H. S. Hassanein, "An architecture for software defined drone networks," in *IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–5.

[5] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. 1st Workshop Hot Topics in Software Defined Networks (HotSDN)*. New York, NY, USA: ACM, 2012, pp. 7–12.

[6] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Commun. Lett.*, vol. 18, no. 8, pp. 1339–1342, Aug 2014.

[7] M. F. Bari *et al.*, "Dynamic controller provisioning in software defined networks," in *Proc. 9th Int. Conf. Netw. Service Manag. (CNSM)*, Oct 2013, pp. 18–25.

[8] A. Alshamrani *et al.*, "Fault tolerant controller placement in distributed SDN environments," in *IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–7.

[9] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Capacity-aware and delay-guaranteed resilient controller placement for software-defined WANs," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 3, pp. 991–1005, Sep. 2018.

[10] M. J. Abdel-Rahman *et al.*, "On stochastic controller placement in software-defined wireless networks," in *IEEE Wireless Commun. and Netw. Conf. (WCNC)*, March 2017, pp. 1–6.

[11] A. Dvir, Y. Haddad, and A. Zilberman, "Wireless controller placement problem," in *15th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan 2018, pp. 1–4.

[12] G. Secinti, P. B. Darian, B. Canberk, and K. R. Chowdhury, "SDNs in the sky: Robust end-to-end connectivity for aerial vehicular networks," *IEEE Commun. Mag.*, vol. 56, no. 1, pp. 16–21, Jan 2018.

[13] M. Lacage and T. R. Henderson, "Yet another network simulator," in *Proc. Workshop NS-2: The IP Network Simulator*, ser. WNS2 '06. New York, NY, USA: ACM, 2006.