Geographic-Region Monitoring by Drones in Adversarial Environments

Ouri Wolfson  
University of Illinois at Chicago  
wolfson@uic.edu

Sushil Jajodia  
George Mason University  
jajodia@gmu.edu

Prabin Giri  
Iowa State University  
pgiri@iastate.edu

Goce Trajcevski  
Iowa State University  
gocet25@iastate.edu

ABSTRACT
We consider surveillance of a geographic region by a collaborative system of drones. The drones assist each other in identifying and managing activities of interest on the ground. We also consider an adversary who can create both genuine and fake activities on the ground. The objective of the adversary is to use fake activities, in order to maximize the response time to genuine activities. We present two collaboration algorithms and analyze their response times, as well as the adversary’s efforts in terms of the number of fake activities required to achieve a certain response time.

CCS CONCEPTS
• Networks → Network algorithms; • Security and privacy;

KEYWORDS
Drones monitoring, Cooperative Drones, Fake events

1 INTRODUCTION AND MOTIVATION
Continuous advances in mechatronics, miniaturization of cameras and other sensing and computing devices, and communication and networking have boosted the application potential of Unmanned Aerial Vehicles (UAVs) in multiple problem domains [2]. As a consequence, in the recent years (UAVs) have experienced extreme popularity gains both in terms of research as well as use in various civilian and military applications [3], spanning through domains such as geographic region monitoring and surveying, search and rescue mission, transport, target detection, etc. [4–7].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGSPATIAL '21, November 2–5, 2021, Beijing, China  
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 978-1-4503-8664-7/21/11 ... $15.00  
https://doi.org/10.1145/3474717.3484216
between the extra and the cell requesting reinforcement. CPH’s whose worst case response time is $m - 1$ (i.e. roughly a square root of the worst case response time of the Naive algorithm), and the number of fake activities required to achieve this response time is also $m$. Using a constant number of fake-events an adversary can only produce a constant response time in CPH.

In summary, this paper makes the following contributions: 1. Introduces a new model that captures collaboration of drones in an adversarial context (Sec. 2); 2. Introduces two drone-collaboration algorithms, Naive (Sec. 3) and CPH (Sec. 4), and analyzes their response time and the cost (in terms of fake activities) to an adversary to achieve a given response time.

2 MODEL

Given a connected geographic region, we partition it into a set $S$ of $m$ of square cells. The length of a square side is $\alpha$. We require that the set $S$ of cells is connected in the following sense. Define a dual graph $G_D(S)$ of a set of cells $S$ as follows: (1) the vertices of $G_D(S)$ represent the cells in $S$; (2) an edge exists in $G_D(S)$ between a pair of vertices if the respective cells share an entire side. We call the cells in the set $S$ connected if its dual graph $G_D(S)$ is connected.

A set of $m + 1$ identical drones is assigned to cover the region. The geographic region and the set of drones is called a drone-system, or a system for short, and is denoted $S(m)$.

At any point in time, each drone in the system is located in at most one cell. However, multiple drones may be located in the same cell. The mapping of each drone to the cell in which it is located at time $t$ is the configuration of the system at time $t$.

When a drone is in a cell we say that the cell is covered by the drone. At any point in time a drone covers at most 1 cell, i.e. the cell in which it is located. A cell may be covered by multiple drones. If at a time $t$ a cell is not covered by any drone, then it is exposed. A configuration is legitimate if it does not contain any cell that is exposed. Configuration changes occur as a result of drones moving from cell to cell and they are constrained to be legitimate.

At any point in time, each cell $c$ has a drone $r(c)$. A drone that is not responsible for any cell has the role of an extra. In this paper we consider systems with a single extra. The transmission range of a drone is $2\sqrt{2}\alpha^2$. This guarantees that each drone can communicate directly with the drones within its cell and its neighboring cells. Furthermore, since the set of cells is connected, a message can be flooded throughout the drone-system. (Flooding of a message occurs by iterative broadcast to all the neighbors, until all nodes in the network have received it.)

We assume that the time to exchange a message between two drones in neighboring cells is negligible. Namely, a typical time for two-way message exchange between drones is 90ms [8], and the drones can fly at between 50mph and 100mph (80km/h – 160km/h) [1]. Thus, we assume that the transmission delays of messages are zero. This means that flooding of a message throughout the drone system takes zero time.

Define a time unit to be the time that it takes a drone to traverse the side of a cell (distance $\alpha$). Furthermore, define the time to be divided into intervals $[1, 2], [2, 3] \ldots$, each of which corresponds to a time unit. Each drone knows the id’s of all the nodes in the system, and their cryptographic keys. Thus, utilizing the instantaneous communication network, each drone knows the current configuration, and the ones planned in the future as well (as we will explain, future configurations may be planned).

The configuration of drones changes as a result of the activities happening in cells. These activities are recognized by the responsible drones of the cells. If an activity is recognized in a cell anytime during a time unit, we say that the cell becomes active at that time unit. Intuitively, a cell becomes active if an event occurs in it (e.g., a fire, or a package ready for pick-up is detected). The cell stays active until it is deactivated by actions of drones; two drones are necessary in a cell in order deactivate a cell. So during each time unit each cell is either red (i.e., active), or white (i.e., inactive). Intuitively, a cell is active if an activity has been detected, but not confirmed. Due to the adversary, 2 drones are necessary for confirmation (because for example, stereo vision is required to determine that the activity is genuine). Once confirmed, the cell is deactivated. And practically, the pickup truck is summoned for the package, or the fire brigade is called. Observe that this means that even though the activity has not ceased, e.g. the fire is still burning, from the surveillance system point of view the cell became inactive since ground forces have been summoned. And if the activity is determined to be fake the cell is also deactivated, since it was determined that ground forces do not need to be summoned.

**ctl2 reinforcement request messages.** Requests for assistance to deactivate a cell are communicated using reinforcement-request messages flooded by the responsible drones. The format of a reinforcement request message is $Control_2 Message(c, t)$ or $ctl2(c, t)$ indicating that the request message is issued at the beginning of time unit $t$ by the responsible drone $r(c)$ that is covering a cell $c$, where $c$ must satisfy following conditions:
- $c$ does not have an outstanding $ctl2$ (an outstanding request is defined below), and
- $c$ is active at the beginning of time unit $t$, and
- $r(c)$ is alone in $c$ at the beginning of time unit $t$.

The message $ctl2(c, t)$ is a reinforcement request. Intuitively, it means that $r(c)$ asks for reinforcement of $c$ by the extra. The $ctl2(c, t)$ request remains outstanding until an extra drone $d$ arrives and stays in the cell $c$ for a whole a time unit; then the extra $d$ is also called the reinforcement drone, and we say that it services the outstanding $ctl2$ request. Servicing of a request $deactivates$ the corresponding cell. Observe that if a reinforcement drone arrives in the middle of time unit 10, then it must stay until the end of the 11th time unit in order for the $ctl2$ message to be considered serviced, and for the cell to be deactivated.

At the end of the time unit when a cell $c$ is deactivated, the extra in $c$ also becomes a floater, i.e., available to satisfy new $ctl2$ requests.

After the $ctl2(c, t)$ request is issued, a floater $f$ becomes allocated to the request, and it keeps the “allocated” role until the cell $c$ is reinforced. Thus, an extra is either a floater, or an allocated, or a reinforcement drone.

**Observation:** The number of active cells during a time unit $t$ may be higher than the number of $ctl2$ messages issued during $t$. The reason is that active cells persist until the cell is reinforced, and reinforcement may happen long after issuance of the $ctl2$ message.
We say that a ctl2 message is satisfied with \( \text{delay} \ q \), or has a \( \text{response-time} \ q \), if the extra drone arrives in the cell \( c \) by the beginning in the \( q^{th} \) time unit after the ctl2 message was issued, but not by the beginning of the time unit \( q - 1 \); and the extra becomes a floater instantaneously at the end of the \( q^{th} \) time unit.

So, for example, if the ctl2 message was issued in the \( 10^{th} \) time unit and a reinforcement drone \( d \) arrives after the beginning and before the end of in the \( 11^{th} \) time unit; then the ctl2 message was satisfied with delay 2, and the cell deactivation occurs at the end of the \( 12^{th} \) time unit; if the reinforcement drone arrives before the end of the \( 10^{th} \) time unit, then the response time is 1, and the cell deactivation occurs at the end of the \( 11^{th} \) time unit.

An algorithm for servicing reinforcement requests makes a decision whenever a reinforcement drone becomes a floater. The decision is which ctl2(c,t) request to service next, and how the extra will move to the cell \( c \). This decision is computational and occurs in 0 time.

Given a system \( S \) and an algorithm \( A \), the maximum possible response time to a ctl2 message is called the response-time of \( A \) on \( S \), and is denoted \( r(S, A) \). Due to the existence of an adversary, discussed below, the objective of \( A \) is to minimize its maximum response-time on \( S \).

Now assume that there is an adversary that knows the system \( S(m) \) and the algorithm used by the system to respond to reinforcement requests\(^1\). The adversary generates fake activations of cells, in order to draw the extra away from some target cell, on which the adversary wishes to perform a genuine activity. Cell activations are genuine if they are not fake. Continuing the color analogy, a cell during a time unit may be red, white or brown (i.e. fake-active).

A single drone can identify that the cell is active, but is color blind and cannot distinguish between the red and brown colors – i.e., it cannot by itself distinguish whether the activity is genuine or fake. Given fake activations, we do not care about the response time to ctl2 requests that are issued in response to fake activations. Formally, the response time of an algorithm (defined above in the absence of fake activations) is the maximum response time to a ctl2 message issued by a genuinely active (i.e. red) cell.

The objective of any algorithm is to minimize the maximum response time, and to maximize the number of fake activations that are necessary for an adversary to increase the response time, and in particular to achieve the maximum response time.

3 NAIVE ALGORITHM

In this section we introduce the Naive algorithm for servicing reinforcement requests. It is called Naive because it behaves in a straightforward manner, in the following sense. A floater services a request by simply flying to the cell from which the request was issued via the shortest path (i.e. along a straight line). Since the objective of any algorithm is to minimize the worst case response time, the floater always picks the oldest request to be serviced next. And if multiple requests are oldest, it picks the closest one (also for minimizing the response time); and if there are multiple requests that are oldest and closest, it picks one of these randomly.

We now discuss some properties of the Naive algorithm. Let a line geographic region (lgr) be a set of cells arranged in a single layer, where each cell has at most 2 neighbors, a right and a left one, but no up, nor down, nor diagonal neighbors.

**Theorem 3.1.** Given a system \( S(m) \) on a line geographic region, \( r(S(m), \text{Naive}) = \Theta(m^2) \).

Intuitively, the above Theorem indicates that the Naive algorithm’s worst case response time is \( \Theta(m^2) \), where \( m \) is the number of cells in the system. The following theorem addresses the adversary’s power. It indicates that using a linear number of fake activations the adversary can generate the worst case response time.

**Theorem 3.2.** By generating at most \( m - 1 \) fake activations in a system \( S(m) \) using the Naive algorithm on a line geographic region, an adversary can produce a response time \( T(m) = \Theta(m^3) \).

Additionally, the following theorem indicates that if the adversary has only single fake activation, the damage (i.e. the response time) she can produce is linear in the size of the system.

**Theorem 3.3.** Consider a system \( S(m) \) using the Naive algorithm on a lgr with cells numbered 1, ..., \( m \). By generating a single fake activation an adversary can produce a response time of at least \( \max(m-j+1, j-1) \) to a ctl2 message issued from cell \( j \).

4 COORDINATED PATH HOP

In this section we introduce a novel algorithm for servicing reinforcement requests called Coordinated Path Hop (CPH), analyze its properties and show that it is superior to the Naive algorithm.

![Figure 1: Illustration of the CPH algorithm](image)

Observe that the Naive algorithm does not make use of the \( m \) drones that are responsible for the cells. More specifically, assuming a single outstanding ctl2 reinforcement-request, none of the \( m \) drones that are responsible for cells participates in satisfying the request. In this section we introduce an algorithm that uses these drones to reduce the response time of a request.

Assume that the extra drone, denoted \( d_g \), is in cell \( g \) and it is allocated to respond to a ctl2(c,t) request. Regardless of the locations of the extra \( d_g \) and the destination cell \( c \), the time it takes to reinforce destination cell \( c \) can be reduced to a single time unit. Intuitively, this can be done by having the allocated drone from \( g \) fly across a single cell-border, rather than flying all the way to \( c \). Furthermore, all the drones in cells forming a path \( p \) from cell \( g \) to cell \( c \) traverse to the next cell in \( p \), including the drone \( d_c \) in the cell before \( c \); \( d_c \) traverses to \( c \), and becomes the reinforcement drone in \( c \). This scenario is illustrated in Fig. 1. If the extra in cell \( g \) is allocated to

\(1\) The algorithms introduced in this paper use randomization, and although the adversary knows the algorithm, she does not know a priori any result of coin tosses of the algorithm.
reinforce cell \( c \), then each drone responsible for a cell along the path from \( g \) to \( c \) moves to the next cell along the path; all the moves occur simultaneously. The drone responsible for the cell before \( c \) is finally providing the reinforcement.

More precisely, consider a time unit \( t' \) at the beginning of which the set of outstanding \( ctl2 \) messages is not empty, and the extra drone \( d_y \) is in cell \( g \) as a floater. Then the extra selects the oldest, or one of the oldest \( ctl2 \) requests, \( ctl2(c,t) \), to service next. And the extra broadcasts \( c \) throughout the network, so that all the drones can calculate a path \( p \) from \( g \) to \( c \). Then, the drones along the path \( p \) move concurrently to the next cell on the path as described above.

**Algorithm 1: The Coordinated Path Hop (CPH) algorithm**

**input:** A drone system \( S(m) \); time unit \( t' \) at the beginning of which the set of outstanding \( ctl2 \) requests is not empty and the extra drone \( d_y \) is a floater - the algorithm is executed during \( t' \); cell \( g \) in which \( d_y \) is located at time unit \( t' \)

1. From the set of outstanding \( ctl2 \) requests \( d_y \) selects the oldest; if multiple \( ctl2 \) requests are oldest, it select one randomly. Denote the selected request \( ctl2(c, t) \), and broadcast its selection throughout the network.
2. Construct a path \( p = \{p_1, p_2, \ldots, p_n\} \) in \( G_D(S) \), and direct it from \( g \) to \( c \) for \( n \) cells.
3. At the end of time unit \( t' \) the following hops occur instantaneously:
   - each responsible drone in cell \( p_i \) hops to cell \( p_{i+1} \) for \( i = 2, \ldots, n - 1 \)
   - allocated drone \( d_y \) hops to \( p_2 \)
4. At the beginning of time unit \( t' + 1 \):
   - assign each newly arrived drone in \( p_i \), \( i \in \{2, \ldots, n - 1\} \) as responsible
   - assign the newly arrived drone in \( p_n \) (cell \( c \)) as reinforcement

Observe that since all the drones use the same algorithm, a unique path is calculated by all the drones. The drones which are not along that unique path, stop their participation in the algorithm. In addition, we note that in line 3 of Algorithm 1 concurrent hops occur. As a consequence of the model (i.e., a drone can travel the length of a cell in one time unit), the preparation for each hop, i.e., positioning of the responsible drone on the border with the next cell, can begin and complete within time unit \( t' \).

Observe that because hops occur instantaneously at the end of a time unit, if a system uses CPH to respond to \( ctl2 \) requests, then configuration changes occur instantaneously in between time units. This means that during a whole time unit each drone stays in a single cell, i.e., drones do not cross cell boundaries in the middle of a time unit. The execution of CPH starting at time unit \( t' \) is completed when the reinforcement arrives at its destination, i.e., at the time unit \( t' + 1 \).

The main difference between the CPH and the Naïve algorithms is in the way the extra moves and the reinforcement arrives to the cell selected for service. Furthermore, when selecting which \( ctl2(c,t) \) request to service, the Naïve algorithm considers the distance to \( c \), whereas CPH does not do so; it picks a random request from the set of oldest \( ctl2 \) requests. Obviously, the reason for this is that under CPH a reinforcement can “reach” any cell in a single time unit, regardless of the distance of the extra to it.

The following analysis of CPH is applicable to any geographic region, not necessarily an lgr. The next theorem indicates that the maximum response time of the CPH algorithm is bounded by the number of cells in the system.

**Theorem 4.1.** Given a system \( S(m), r(S(m), CPH) \leq m - 1 \).

In other words, compared to the Naive algorithm, CPH cuts the worst-case response time from \( O(m^2) \) to \( m - 1 \).

Now we consider the adversary, and show that a response time of \( m - 1 \) is achievable. Furthermore, we show that achieving it requires \( m - 1 \) fake activations.

**Theorem 4.2.** Given a system \( S(m) \), the number of fake activations that is necessary to produce a response time of \( m - 1 \) in the CPH algorithm is \( O(m) \). Furthermore, \( O(m) \) fake activations are sufficient to produce a response time of \( m - 1 \).

Theorem 4.2 indicates that even though the worst case response time of CPH is \( O(m) \), the adversary cannot guarantee this response time using a constant number of fake cell-activations, as she does in the Naive algorithm (cf. Theorem 3.3); she will need a linear (in the number of cells) number of fake activations to do so.

5 CONCLUDING REMARKS

We addressed the problem of jointly reducing the worst-case response times and maximizing the adversaries’ efforts to produce these. We do so in the setting of drone-based surveillance of a geographic region of interest. We considered two algorithms – Naïve and CPH – and demonstrated that: (a) the worst-case response times for the Naïve and CPH algorithms are \( O(m^2) \) and \( O(m) \) respectively; and (b) an adversary needs a constant number of fake activations (i.e., resources) to produce an \( O(m) \) response time in the Naive algorithm, vs. an \( O(m) \) number of fake activations to produce the same response time in CPH; and (c) in the Naive algorithm, by using \( O(m) \) fake activations the adversary can produce on \( O(m^2) \) response time, whereas such a response time is impossible in CPH regardless of the number of fake activations.

**Acknowledgments:** Research in part supported by ONR grants N00014-18-1-2670 and N00014-20-1-2407, ARO grant W911NF-13-1-0421, and NSF SWIFT grant 2030249.

**REFERENCES**