# Chapter 1

# Streaming Changes of Centrality of UAV Networks

Contributed by Joshua Huseman

## 1.1 Introduction

The Dronology project is a system for controlling a fleet of multiple Unmanned Aerial Vehicles (UAVs) in formation[2]. With the current iteration of the project, communication is established as a direct serial over Radio Frequency (RF) link between the Ground Control Station (GCS) and each UAV, necessitating a separate communication antenna connected to the GCS computer for each UAV it controls. As the project progresses and expands to large fleets of UAVs, it is quickly becoming necessary to de-centralize communication between the UAVs in a fleet. The planned strategy is to use Wi-Fi direct links between UAVs to create a mesh network, with certain UAVs functioning as intermediary "hubs" relaying instructions between the GCS and other UAVs. While a simple solution would be to simply designate that each UAV should act as a "hub" and broadcast messages to the entire network, this introduces many inefficiencies, such as unnecessary overhead and bandwidth usage when passing messages that are not needed in certain parts of the network. It is necessary to create an algorithm to determine an optimal layout of the network, in order to establish an efficient line of communication between UAVs, as well as a heirarchy of decision making for issues like collision avoidance and route planning.

Further, because the fleet of UAVs is constantly in motion, the shape of this network is likely to change frequently, making evolution of the network layout very important. It will greatly improve performance of the system if the layout of the network can be analyzed incrementally, rather than re-building the network "from scratch" every time a UAV moves.

It will also greatly reduce strain on the network to distribute the computation of this algorithm across the UAVs in the network, rather than on the central GCS, as information about the optimal network layout will not have to be distributed through the entire network after every change. This will allow more bandwidth to be used for faster communication of important information through the network, rather than consumed by metadata about the network.

## 1.2 The Problem as a Graph

There are two aspects of this problem which can be expressed as graphs: the graph of possible data links connecting the UAVs and the graph of the network generated for communication between the

UAVs. The first graph would be the input to the algorithm developed here, which would generate the second of these two graphs. The possible network connections can be expressed as an undirected weighted graph. The nodes of the graph represent UAVs, with one specific node representing the GCS, and the edges represent available data links between the UAVs (and the GCS), with the weights of the edges representing the "cost" of the link. This "cost" will be a variable that might change based on the exact implementation of the algorithm, but it can be expected that the "cost" will increase as distance increases or as signal strength or speed decreases. The generated network should be a directed rooted tree, or arborescence, with the nodes representing UAVs (and the GCS) and the edges representing the chosen links to be used by the network, with the direction of the edges going from a "hub" UAV to the connected UAV it is directing. This generated arborescence must also be a subset of the original graph, with all of the same nodes, and the edges being a subset of the edges from the original graph (with directions rather than weights). Any nodes with an edge going away from them represent "hub" nodes, and the root of the arborescence must be the node representing the GCS.

## 1.3 Some Realistic Data Sets

The data for this problem can be generated quite easily. When implementing the algorithm in practice, the signal strength of data connections can be obtained from each UAV, and converted to "cost" values for the weight of the corresponding edge in the graph. As it is difficult to run large-scale tests with actual UAVs, it is necessary to generate larger test data sets procedurally. This can be achieved by running a simple simulation of the fleet of UAVs, likely with a simple random walk, or diffusion, algorithm determining the positions of the UAVs. Then the links between nodes can be created with a weight proportional to the distance between nodes (with a maximum threshold distance for an edge). This allows for creation of hypothetical datasets much larger than any measured data we have available. Some of the largest tests which the Dronology project has done with physical UAVs so far have been 5 or 6 UAVs, but the project's eventual goal is to support hundreds or thousands of UAVs in a fleet[2]. There will also be a variety of fleets, some spread out over large areas, while others are in dense groups, necessitating a variety of test graphs, both sparse and dense, to properly represent and test as many situations as possible.

## 1.4 Network Centrality-A Key Graph Kernel

The problem can be solved by first calculating a centrality measure for each UAV, then using that centrality measure as a priority for constructing an arborescence with all of the nodes of the graph. The centrality measure may use many different algorithms, but Closeness Centrality seems to be the most promising option so far. This is calculated for all nodes using Algorithm 1:

Algorithm 1 has time complexity of $O(n^2)$, n=num nodes, multiplied by the time complexity of the shortest path algorithm. The most efficient implementations of the commonly-used Dijkstra's algorithm is $O(e+n*log(n))$, e=num edges [1], making the time complexity $O(e*n^2+n^3*log(n))$. The space complexity is $O(n)$, added to the space complexity of the shortest path algorithm. Dijkstra's algorithm's space complexity is typically $O(n)$ [1], so the space complexity of the algorithm is still just $O(n)$.

So far, the simplest (but not very efficient) algorithm for building the network seems to be Algorithm 2:

Algorithm 2 has a worst-case time complexity of $O(n^2)$, added to the time complexity of Algorithm 1 and the time complexity of the sorting algorithm used to determine the order to use

**Algorithm 1** Closeness Centrality

```
nodes: set of nodes in graph

create number list farness
create number list closeness
for x in nodes:
    farness[x] = 0
    for y in nodes:
        shortest_len = path_length(shortest_path(x,y))
        farness[x] = farness[x] + shortest_len
    closeness[x] = len(nodes)/farness[x]
```

**Algorithm 2** Network Building Algorithm

```
nodes: set of nodes in graph
gcs_root_node: node corresponding to the GCS

function sort_by_centrality(nodes):
    calculate centrality using Algorithm 1
    return nodes sorted by centrality

remaining_nodes = sort_by_centrality(nodes)
create node set hubs
hubs.append(gcs_root_node)

function set_hub(node):
    connected_nodes = get_connected_nodes(highest_node)
    remaining_nodes.remove(connected_nodes)
    connected_nodes.remove(hubs)
    hubs.append(node)
    for x in connected_nodes:
        node.children.append(x)

while len(remaining_nodes)>0:
    highest_node = remaining_nodes[0]
    connected_nodes = get_connected_nodes(highest_node)
    closest_dist = infinity
    for x in hubs:
        this_path = shortest_path(highest_node,x)
        this_dist = path_length(this_path)
        if this_dist < closest_dist:
            closest_hub = x
            closest_path = this_path
            closest_dist = this_dist
    for y in closest_path:
        set_hub(y)
```

the nodes. Since most sorting algorithms are faster than $O(n^2)$, the overall time complexity is the same as that of Algorithm 1, $O(e * n^2 + n^3 * log(n))$. The space complexity of algorithm 2 is $O(n)$, added to the space complexity of the sorting algorithm used. As many sorting algorithms use less space than $O(n)$, this leaves the overall space complexity at $O(n)$.

Analysis of the performance of these algorithms can be done using a number of metrics, including number of hubs used and average cost of connections between UAVs and the GCS. When making variations to the centrality metric used or optimizations to the algorithm, these metrics can be used to confirm that performance is not degraded significantly.

## 1.5 Prior and Related Work

There has been previous research into using a more common ad-hoc networking scheme for communication between UAVs in a fleet[3]. These ad-hoc networks use a decentralized mesh-based structure for communication, in a similar way to what is proposed here. While this allows for decentralized communication around the network, most solutions still use a centralized approach for authority, forcing all decisions to be made by a central Ground Control Station. Creating a heirarchy of authority between the UAVs allows the UAVs to take care of smaller decisions, such as small-scale collision avoidance, without contacting the GCS directly. This decentralized decision-making allows for lower latency in time-critical applications like collision avoidance, saving valuable time for the UAVs to react to one another, along with freeing up bandwidth in the parts of the network closer to the GCS, leaving room for larger instructions to UAVs, without having to micro-manage the small details.

# Bibliography

[1] Mo Chen, Rezaul Alam Chowdhury, Vijaya Ramachandran, David Lan Roche, and Lingling Tong. Priority queues and dijkstra's algorithm, October 2007.

[2] Jane Cleland-Huang and Michael Vierhauser. Dronology - https://dronology.info/.

[3] D. L. Gu, G. Pei, H. Ly, M. Gerla, B. Zhang, and X. Hong. Uav aided intelligent routing for ad-hoc wireless network in single-area theater. In *2000 IEEE Wireless Communications and Networking Conference. Conference Record (Cat. No.00TH8540)*, volume 3, pages 1220–1225 vol.3, Sept 2000.