Caching techniques for ad hoc drone networks

Diploma Thesis

By

Xristos Theologou

Department of Electrical and Computer Engineering
University of Thessaly

A dissertation submitted to the University of Thessaly in accordance with the requirements of the Diploma degree in the department of Electrical and Computer Engineering.

September 2019

Περίληψη

Τα τελευταία χρόνια υπάρχει μεγάλη τάση ενασχόλησης και ανάπτυξης τεχνολογιών σε μή επανδρωμένα αεροσκάφη (drones), και ακόμα μεγαλύτερη αναμένεται μέσα στα επόμενα χρόνια τόσο για εμπορικές όσο και για προσωπικές εφαρμογές. Οί χρήσεις τους ποικίλουν, ωστόσο μία απο τις πιό ενδιαφέρουσες είναι η ανάπτυξη ασύρματων δικτύων. Ένα από τα πιό σημαντικά σχεδιαστικά ζητήματα είναι η επίτευξη της γρήγορης και αξιόπιστης επικοινωνίας μεταξύ των μή επανδρωμένων αεροσκαφών καθώς και της διαχείρισης της κατανάλωσης της ενέργειάς τους.

Η διάχυση της πληροφορίας σε ενα υβριδικό ασύρματο δίκτυο (cellular και ad hoc) αποτελούμενο από μή επανδρωμένα αεροσκάφη είναι σημαντική πρόκληση εξαιτίας των ιδιαιτεροτήτων της κίνησης και των απαιτήσεων των μη επανδρωμένων αεροσκαφών. Το δίκτυο που δημιουργείται αποτελείται από κόμβους, όπου κάθε κόμβος αντιστοιχίζεται σε ένα μη επανδρωμένο αεροσκάφος, και υπάγεται στην κατηγορία ενός MESH δικτύου καθώς έχει τις ιδιότητες να μεταβάλλεται η τοπολογία του συνεχώς, να είνα αυτορυθμιζόμενο και να έχει περιορισμένη ενέργεια. Τα κυριότερα προβλήματα που εμφανίζονται είναι η εύρεση συντομότερου μονοπατιού καθώς και η δημιουργία ενός ενεργειακά αποδοτικού δικτύου.

Η προσφορά της παρούσας διπλωματικής εργασίας είναι η σχεδίαση ενός λογισμικού προσομοίωσης δικτύου από drone, αλλά και ο σχεδιασμός ενός αλγορίθμου caching, ο οποίος ανακαλύπτει κάθε φόρα το συντομότερο αλλά και το ενεργειακά καλύτερο μονοπάτι για την εύρεση της πληροφορίας. Το αποτέλεσμα της εφαρμογής του αλγορίθμου είναι η καλύτερη αξιοποίηση της ενέργειας των μη επανδρωμένων αεροσκαφών, που έχει σαν συνέπεια την μεγαλύτερη βιωσιμότητα του δικτύου, καθώς και την γρηγορότερη εξυπηρέτηση των αιτήσεων εντός του δικτύου ενός μη επανδρωμένου αεροσκάφους.

Abstract

In recent years, there has been a great tendency to deploy and develop technologies in drones, and even more is expected in the coming years for both commercial and personal applications. Their uses vary, but one of the most interesting is the development of wireless networks. One of the most important design issues is to achieve fast and reliable communication between unmanned aircraft as well as managing their energy consumption.

Dissemination of information to a hybrid wireless network (cellular and ad hoc) consisting of unmanned aircraft is a major challenge due to the specificity of the movement and the requirements of unmanned aircraft. The network created consists of clusters, where each node corresponds to an unmanned aircraft, and falls into the category of a MESH network as it has the ability to continuously vary its topology, be self-regulated and have limited energy. The main problems that arise are finding a shorter path and creating an energy efficient network.

The purpose of this thesis is to design a drone network simulation software, as well as to design a caching algorithm, which explores each time the fastest and most efficient way to find information. The result of applying the algorithm is to make the best use of the energy of unmanned aircraft, which results in greater network viability as well as faster handling of requests within the network of an unmanned aircraft.

Dedication and acknowledgements

In the first place I want to thank my lead supervisor, Dr. Dimitrios Katsaros and Dr. Athanasios Korakis for their assistance and guidance for the completion of this work. I would also like to thank my friends and my family that were by my side all these years.

To my family and friends.

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: .... DATE: ....

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction to UAV Ad-Hoc Technologies

Over the past few years, unmanned aerial vehicles (UAVs), also known as drones, has been one of the main research subjects, owing to their relatively low cost, high mobility, autonomy and board range of application domains. Indeed, UAVs have been adopted to a various of applications such as military, agriculture, wireless sensors and monitoring, telecommunications and QoS improvement, delivery of supplies, rescue operations and so on. Most of these applications are developing a wireless ad hoc or mesh network.

A wireless mobile ad hoc network is a decentralised type of wireless network. They are self-configuring, dynamic networks in which nodes are free to move. Furthermore, the network is ad hoc because it does not rely on a pre-existing infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks. Instead, each node participates in routing by forwarding data for other nodes. The way packets travel is similar to the wired Internet. Data will hop from one device to another until it reaches its destination. The communication is achieved with dynamic routing algorithms implemented in each device. To implement such as algorithms, each device needs to know routing information to other devices in the network. Each device then determines what to do with the data.

In the comming years, wireless Ad Hoc networks of drones, also known as flying Ad Hoc Network (FANETs) [3, 14] will be of great use due to the many capabilities they offer. The advantages of a FANET construction can be summarized as follows:

- Cost: The acquisition and maintenance cost of small UAVs is much lower than the cost of a large UAV

- Scalability: The usage of large UAV enables only limited amount of coverage increases. However, multi-UAV systems can extend the scalability of the operation easily.
- Speed-up: It is shown that the missions can be completed faster with a higher number of UAVs
- Survivability: If the UAV fails in a mission which is operated by one UAV, the mission cannot proceed. However, if a UAV goes off in a multi-UAV system, the operation can survive with the other UAVs.

Although there are several advantages in such types of systems, it has also unique challenges. The most significant problem in such types of networks is the communication and the energy consumption. Wireless networks are constantly improving, adopting new protocols and technologies. such as WiFi 802.11, Zigbee which is used to to create small personal area with low-power digital radios or LoRa which is a spread spectrum modulation technique with low-power consumption. In order to cope with the communication problem, algorithms and techniques like caching or shortest path exploration [16] are being used to make faster the communication.

So, this thesis builds an energy-efficient caching algorithm, in order to decrease the number of hops inside the network and to make a better energy management. In order to test the algorithm, a simulator engine was developed to define the drones properties into the network. This algorithm can work above other wireless technologies, as we mention above.

The chapters included in this thesis are organized in three parts, as can be seen in table

Organization of Chapters		
Chapters	Parts	Themes
Chapter 1	Introduction	Introduction to UAV Ad-Hoc Tecnhologies
Chapter 2	Background	Unmanned Aerial Systems Flying Ad-Hoc Networks Networking Architectures Routing Protocols Caching
Chapter 3	Implementation	System Model NDR Simulator Energy Efficient Caching
Chapter 4	Research Results	Experiments and Results
Chapter 5	Conclusion	Conclusion and Future Work

Table 1.1: Thesis Chapters

Chapter 2

Background

2.1 Unmanned Aerial Systems (UAS)

In the last two decades, as a result of the rapid technological advancement in computation, sensor, communication and networking technologies, Unmanned Aerial Vehicles (UAVs) promise new application areas for military and civilian areas such as, relay for ad-hoc networks, search and rescue operations [19], electronic attacks in hostile areas, ground target detection and tracking, automatic forest fire monitoring and measurement [8], wind estimation, disaster monitoring , remote sensing [22], airspeed calibration, agricultural remote sensing [22], etc. It is a relatively easy task to use UAVs in an Unmanned Aerial System (UAS) for increasing communication range and data aggregation capability of nodes in a system. In case of infrastructure less places, such as enemy territories or natural disaster areas, there is an immediate need to build a network between teams, and a fast deployed UAV can be an acceptable solution as a relay node.

In single-UAV applications, each ground node can easily communicate with the UAV, and thus establish a star topology network structure where the UAV is at the center of the star. By using this topology, a ground node can indirectly communicate with others over the UAV. However, single UAV systems have some challenging issues in peer-to-peer communication such as increasing transmission range, sending more data, and minimizing any interference. A solution to these problems is the use of high gain directional antennas instead of omnidirectional antennas. Undoubtedly, this leads to a limited improvement in the performance of UAS, and this is not satisfactory.

To provide a longer presence over the theatre of operation in order to carry powerful processors, sensors and communication hardware, a large UAV is preferred in single vehicle UASs, in which setting up a communication environment is easy. However, they are not only heavy and large but also pose a significant danger to human life and property in case of a failure. Moreover, they are expensive, and failure of a UAV can have a high cost.

On the other hand, technological advancement of electronics and sensor technologies have decreased the production cost of UAVs, and low cost mini-UAVs are becoming more and more popular in both academic areas and practical applications. They are small in size and weight; therefore, they pose little or no threat to human life and buildings/properties. In addition, they can be reused for various types of applications. It is difficult to detect and track UAVs while they are flying, due to their size. They can fly at much lower altitudes, can be easily transported to different areas, and can be launched by an individual in any kind of terrain without a runway or a specific launching device. However, mini-UAVs have restricted capabilities such as power, sensing, communication, and computation; however, the use of a team with mini-UAVs, as multiple UAVs, improves the capability and capacity of UASs and provides a flexible platform for a variety of applications. Usage of multi-UAV systems has significant advantages over single UAV systems:

- Especially in search missions, the usage of a number of UAVs can parallelize individual tasks thus decreasing the completion time of a mission.
- In a single UAV system, if the UAV or a sensor/hardware fails, the UAV should return to the base. However, in multi-UAV systems, other UAVs can share tasks among themselves and this increases the fault tolerance of the system.

On the other hand, these advantages and improvements have a challenging issue: efficient communication and coordination of UAVs in the team. Therefore, a multi-UAV system needs some required hardware and a well-defined networking model whose communication/network layer is depicted in Figure 2.1.

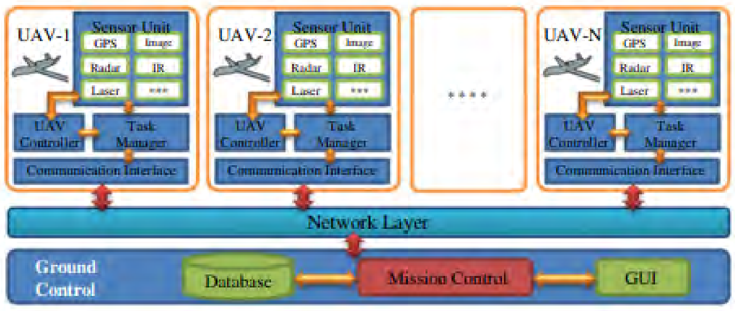


Figure 2.1. A multi-UAV system's processing units and communication/network layer

2.2 FANET

Flying Ad Hoc Networks (FANETs) belong as a subclass of Mobile Ad Hoc Networks (MANETs) and Vehicular Ad Hoc Networks (VANETs), for that reason, common technologies and strategies could be used for data delivery [3]. However, for the proper functionality of data delivery, we have to reconsider and built techniques and architectures that are specifically adapted to FANETs, e.g., high mobility, sparse deployment, continuously changing network topology, intermittently connected communication links.

Over the last decades, many researchers have explicitly studied the communication algorithms and routing protocols in Mobile Ad Hoc Networks (MANETs) and Vehicular Ad Hoc Networks (VANETs), respectively. However, due to the high mobility, sparse deployment, drastically changing network topology, intermittently connected communication links, and intentional jamming and disruption, those mechanisms that were specifically designed for MANETs or VANETs cannot be directly applied in FANETs. In other words, routing demands of FANETs go beyond the needs of MANETs and VANETs [11].

In this section, we categorize and analyze existing networking architectures, routing protocols, caching techniques and algorithms in ad-hoc architectures and other approaches.

2.2.1 Networking Architectures

Recently researchers are developing their work under a concrete network architecture. As presented in [17] the main idea is divided into a centralized and a decentralized architecture. Centralized network topology involves a data center (DC) or a base station in the ground (GS), which is working as a central node and every drone is connected directly with it. 3.9(a)

The decentralized architecture is categorized into three more topologies. The first one is a UAV ad-hoc network topology, in which every drone contributes to the data forwarding process in order to service all the other drones of the network, as shown in Figure 2.2(b). The next configuration is the multi-group UAV ad-hoc network, in which drones within a group, create an ad-hoc network and the backbone drone of each network group, communicates with the central node to the ground 3.9(b). Another network topology is the multi-layer UAV ad-hoc network, which a mediator drone connects different ad-hoc networking groups of UAVs. The Figure 2.2(d) shows, that the information is transmitted from the backbone drones of the networking groups, and every backbone drone is connected to each other.

To sum up, the decentralized communication topology is appropriate for connecting multiple drones in a network. On the one hand, decentralized architecture come up with an extended coverage for data transmission, but on the other hand, comes up with the big problem of data path finding.

2.2.2 Routing Protocols

The categorization of routing protocols is divided into four main categories, static, proactive, reactive and hybrid.

- In static routing, the destination of a packet is computed and saved in tables when a task starts, and these tables cannot be computed or change during the delivery process.[10] This protocol is lightweight but cannot be used in dynamically changing environment like an Ad Hoc Network of drones. Drones are used as packet carriers and transfer data from source to destination.

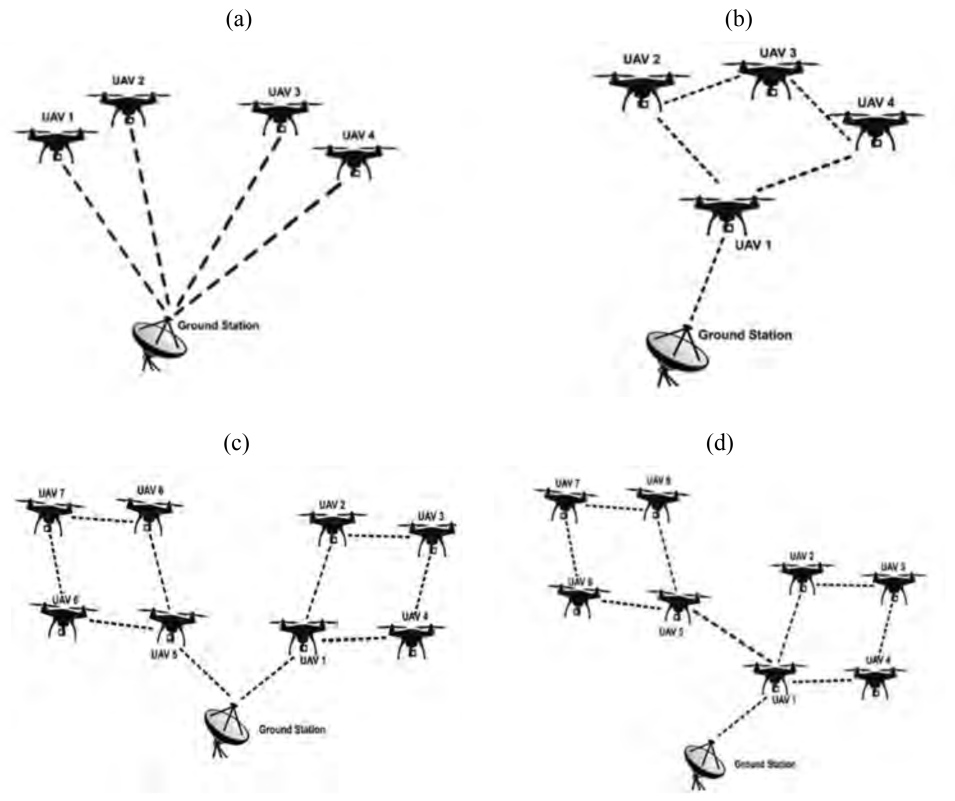


Figure 2.2. (a) A centralized architecture. (b) Unmanned aerial vehicle (UAV) ad-hoc network. (c) Multi-group UAV ad-hoc network. (d) Multi-layer UAV ad-hoc network.

- In the proactive approach we have periodically refreshed and shared routing tables among the drones, resulting in the availability of routing paths between every pair of drones in the network. Thus, the routing paths can be selected to transmit data packets immediately without delay. The main advantage of proactive routing is that it contains the latest information of the routes. However, a large amount of control packets are needed to keep the routing tables up-to-date. In [1], a directional optimized link state routing protocol (DOLSR) is proposed to minimize the number of multi-point relays in FANET, where each drone is equipped with directional and omni-directional antennas.

- Reactive routing protocols ,also known as on-demand routing, can be used to find a path from source to destination when data packets need to be sent. The main profit of the approach is communication overhead reduction as we do not exchange periodic messages. Dynamic source routing (DSR) [15] is a classic reactive routing protocol for multi-hop wireless mesh networks. In DSR, a source node floods a route request packet throughout the network. When the route request packet reaches the destination, the destination replies a route reply packet to source node. In addition, each node can quickly learn the routes of other nodes by aggressively overhearing on-flying packets and caching the piggybacked route information in its routing table.

- Hybrid protocols use both proactive and reactive models and is used to overcome the control message overhead of proactive and the high end-to-end delay of reactive routing. The [12] proposes a zone routing protocol (ZRP), which is a hybrid routing framework suitable for a wide variety of mobile ad-hoc networks, especially those with large network spans and diverse mobility patterns. Each node proactively maintains routes within a local region, also referred to as the routing zone. Knowledge of the routing zone topology is leveraged by the ZRP to improve the efficiency of a globally reactive route query/reply mechanism.

2.2.3 Caching

Several studies have addressed content caching and content replacement in wireless networks. In an ad hoc network, drones are sending data requests but are also serve requests from other drones ,as a data center would do. Such network architectures explained above in section 2.2.1. So it is wise and important to save data in cache that will either the drone itself will request or someone else will. Most known caching algorithm is LRU which discards the least recently used items first, when the cache memory is full. In contrast to LRU, MRU deletes first the most recently used items. The idea behind MRU is that, when a file is being repeatedly scanned in a reference pattern (Looping Sequential), MRU is the best because we save the data that we are going to use and we remove the data that has been used.

The following section, demonstrates some works that are performing different caching techniques in wireless networks.

- Cooperative Caching

  In [18], distributed caching strategies for ad hoc networks are presented according to which nodes may cache highly popular content that passes by or record the data path and use it to redirect future requests. In [8], a cooperative caching technique is presented and shown to provide better performance than HybridCache. However, the solution that was proposed is based on the formation of an overlay network composed of "mediator" nodes, and it is only fitted to static connected networks with stable links among nodes.These assumptions, along with the significant communication overhead needed to elect "mediator" nodes, make this scheme unsuitable for the mobile environments that we address.

  The work in [9] proposes a complete framework for information retrieval and caching in mobile ad hoc networks, and it is built on an underlying routing protocol and requires the manual setting of a network wide "cooperation zone" parameter. Note that assuming the presence of a routing protocol can prevent the adoption of the scheme in [9] in highly mobile networks, where maintaining network connectivity is either impossible or more communication expensive than the querying/caching process. Furthermore, the need of a manual calibration of the "cooperation zone" makes the scheme hard to configure,because different environments are considered.

- Content Diversity

Mobile nodes in [5] cache data items other than their neighbors to improve data accessibility. In particular, the solution in [5] aims at caching copies of the same content farther than a given number of hops. However, this scheme requires the maintenance of a consistent state among nodes and is unsuitable for mobile network topologies. Caching different content within a neighborhood is also exploited in [20], where nodes with same mobility patterns and similar interests are grouped together to improve cache hit rate, and in [13], where neighboring mobile nodes implement a cooperative cache replacement strategy. In both works, the caching management is based on instantaneous feedback from the neighboring nodes, which requires additional messages. The estimation of the content presence that we propose, instead, avoids such communication overhead.

- Clustering Caching

  The Zone Cooperative (ZC) [6] , the Cluster Cooperative (CC) [7] and the ECOR [21] protocols attempt to form clusters of nodes based either in geographical proximity or utilizing widely known node clustering algorithms for MANETs [2]. In ZC, mobile nodes belonging to the neighborhood (zone) of a given node form a co-operative cache system for this node since the cost for communication with them is low both in terms of energy consumption and message exchanges. Each node has a cache to store the frequently accessed data items. The data items in the cache satisfy not only the node's own requests, but also the data requests passing through it from other nodes. For a data miss in local cache, the node first searches the data in its zone before forwarding the request to the next node that lies on a path towards the data center. As a part of cache management, a value-based replacement policy based on popularity, distance, size and time-to-live was developed to improve the data accessibility and reduce the local cache miss ratio. Simulations experiments revealed improvements in cache hit ratio and average query latency in comparison with other caching strategies.

  In CC, the authors present a scheme for caching in MANETs. The goal of CC is to reduce the cache discovery overhead and provide better cooperative caching performance. The authors partitions the whole MANET into equal size clusters based on the geographical network proximity. In each cluster, CC dynamically chooses a"super" node as cache state node, to maintain the cluster cache state information of different nodes within its cluster domain. The cache state node is defined as the first node that enters the cluster. The cluster cache state for a node is the list of cached data items along with their time-to-live field. The cache replacement policy is similar to that in ZC. However, the ZC protocol is problematic in terms of communication overhead and energy consumption, because every node that lies on the path towards the data center has to broadcast the packet to nodes in its zone in order to discover if there is a cached copy that satisfies the requested data item. In contrast with the ZC protocol, the CC protocol reveal a smaller overhead in terms of messages exchange between nodes, because the intermediate node (the nodes between a requesting node and the node which holds the requested data) broadcast the packet only to cluster state node. In ECOR, each mobile node forms a cooperation zone (CZ) with mobile nodes in proximity by exchanging messages to share their cached data items in order to minimize bandwidth and energy cost for each data retrieval. When a data request arrives, the node first searches the data in its CZ before forwarding the request to the data center. According to ECOR each node broadcast every modification of the cached data items to nodes that belong to cooperation zone. Each node maintains a cache hint table for the cache information of all nodes in its proximity. However, in ECOR appears a great number of exchanged messages and energy consumption in case of large node density and big cooperation radius.

Criteria	Static	Proactive	Reactive	Hybrid
Main Idea	Static routing table	Table driven	On demand	reactive&proactive
Complexity	Low	Medium	Average	Average
Route	Static	Dynamic	Dynamic	Dynamic
Topology Size	Small	Small	Large	Both Small and Large
Memory Size	High	High	Low	Medium
Fault Tolerant	Absent	Present	Present	Mostly Present
Bandwidth Utilization	Maximum	Minimum	Maximum	Medium
Convergence Time	Fast	Slow	Mostly Fast	Average
Communication Latency	Low	Low	High	High
Mission Failure Rate	High	Low	Low	Very Low
Popularity	Less	Medium	Medium	High

Table 2.1: Comparisons among the Basic routing protocols in FANETs

Chapter 3

Implementation

In this chapter we are going to demonstrate a distributed algorithm for energy efficient management and caching in a flying ad-hoc network (FANET) of drones. The purpose of the Energy Efficient Caching algorithm is to reduce the network's energy consumption by finding the shortest energy efficient path in order to serve a data request. For our work, we build a simulator to run and test the EEC (Energy Efficient Caching) which we will describe in the next section.

3.1 System model

Figure 3.2 shows a flying ad-hoc network. As we discussed above in the section 2.2.1, we have four main networking architectures. We decide to work on a decentralized multi-layer topology 2.2(d) in which the drones are able to fly in a space area with the assumption that they do not exceed their range area. We do not specify flying areas for each drone, but they are free to enter others drone areas or even they can swap areas as shown in Figure 3.1. Therefore, this architecture has a high mobility and constantly changing topology in contrast with other flying ad-hoc architectures like Paparazzi UAVs [4] that describes 5 possible movements of drone inside a certain area, like Stay-At movement, which is hovering to a fixed position. As a result of this pattern mobility, it is impossible to know the next drone position, so every drone must update its position by sending at t time slots a beacon message.

All the drones in the ad hoc network may have wireless interfaces to connect to the wireless infrastructure such as wireless LAN or cellular networks. Each UAV communicates with any other UAV, if they are in a proper distance, with a two-way communication link in order to transmit and receive data. Moreover, every drone has a cache memory to keep the high importance data and in case of cache missing, there is a capability to reach them from a Data Center. We assume that the communication between the Data Center and the drones is possible via a Base Station as we see in Figure 3.2. A communication link with the DC (Data Center) could be accomplished with more than one drone, as we are working to a multi-layer networking architecture. Suppose that we have a data set $D = d_1, d_2, d_3, ..., d_n$ in the Data Center. The drones of our ad-hoc network are interested about these data. A drone has a cache of $x = x_1, x_2, ..., x_i$ where

$$i << n$$

. So either a drone has the most popular data saved in its cache or it has to request them from the network. We also consider that every drone has a battery life limit B_L . Eventually, we assume that as the drone flies and communicates with other neighbour drones, energy consumption is increasing.

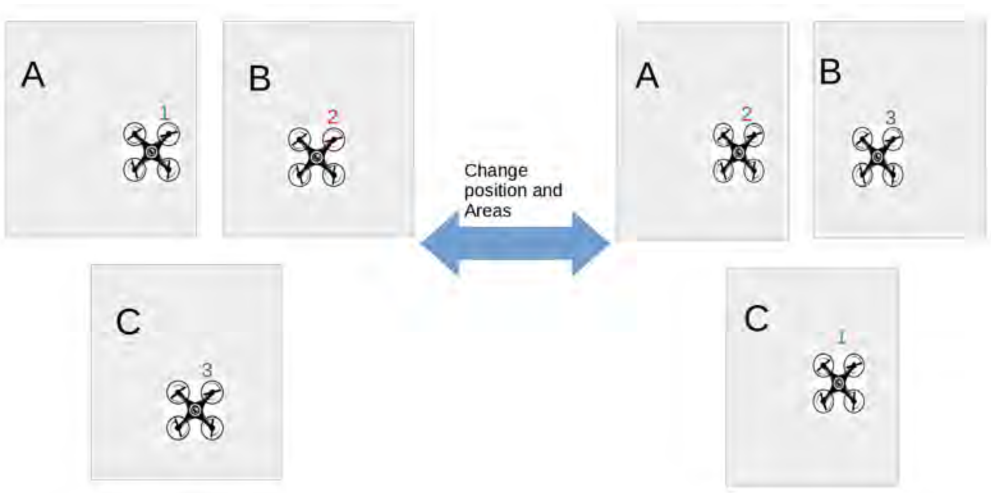


Figure 3.1. The Figure shows the high mobility of a drone able to swap areas.

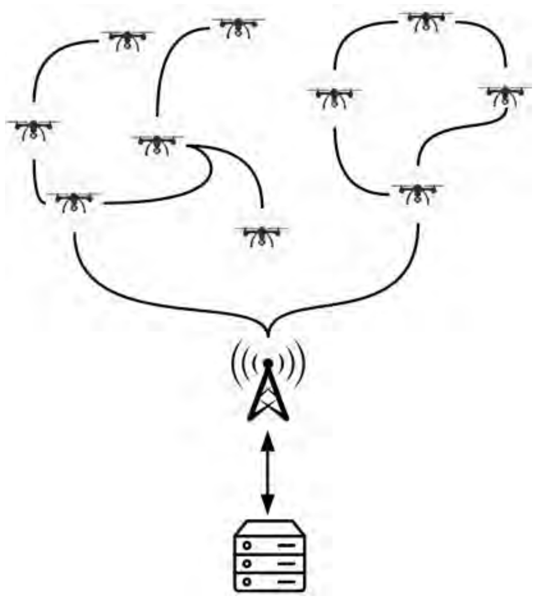


Figure 3.2. The Figure shows the flying ad-how architecture we use.

3.2 Network DRone Simulator

3.2.1 System Architecture

It is of high importance, before going into the Energy Efficient Caching algorithm, to show the overall illustration of the whole system architecture. In Figure 3.3 we see the NDR simulator (Network DRone simulator) components and how they communicate. Under the dashed line we see the back-end components configuration when the simulator starts. The user gives an input to define the number of drones he wish to have in his network. Afterwards, the simulator configures a fully instance of the network. Bellow we analyze in more details the components and how they work.

- Position Controller

  At the begging, Position Controller determines to each drone a random position inside the air space, which the simulator defines, and they exchange beacon messages in order to learn their neighbour. A drone is moving randomly every t seconds. The Position Controller is responsible to determine the next location of each drone. Furthermore, PC (Position Controller) enables the communication between two drones by sending a positive or negative signal to Communication Function and also the communication between a drone and the Data Center. A positive signal will be if their 3D Euclidean distance d is under a threshold distance d_{th} :

$$d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} < d_{th}$$

- Communication Function

  The Communication Function is a component responsible for transmitting and receiving. We have two different types of messages, the beacon and the data message.

  Each drone sends constantly beacon messages in order to inform the network for their position. The beacon message is divided into three parts , as we see in Table **??** . The PacketID is recognized as a sequence which indicates the type of message, in our case the beacon message. The Coordinates, is the part that contains the current position about the drone that send the beacon message. And finally, a drone needs to know the battery remaining energy of its neighbors, so the third part of the message includes the remaining energy. After getting the beacon messages, the simulator transfers the job to Position Controller and PC response with a negative or positive signal.

  The Data message , is that kind of packet that transfers the data items. PacketID has the same role as in beacon message, but here we have different formation as we see in Table 3.1. The second part is the Data, where data item exists and the third and fourth part is the source and destination drone respectively.

Beacon Message		
PacketID	Coordinates	Remaining_Energy

Data Message			
PacketID	Data	Source	Destination

Table 3.1: Beacon and Data message parts

- Cache

  As we said before, every drone has a cache memory. So, the system initialize the memory with values that chosen randomly from data set which exist in our Data Center. We chose to do that in order to avoid the first state in which the cache has no data. As we see in Table 3.2 cache saves a data item and its importance number. When cache manager finds the data item in cache, we have a cache hit and cache manager increase by one the importance number of the item.

Cache Memory	
Data Items	Importance Number
d_1	4
d_2	5
d_3	3

Table 3.2: Cache memory instance

- Energy Consumption Controller

All the drones of the users network starts with full battery, so that is Battery = 100 in our simulator. ECC (Energy Consumption Controller) component controls the energy of every drone. The energy of the drone is reduced when it is driven from one place to another. The equation for this reduction is :

$$RE_{movement} = \frac{d_{current} - d_{previous}}{UL} * e_{drone}$$

where the $RE_{movement}$ the remaining energy from the movement, UL is the unit length we choose and e_{drone} is the Battery counter. Energy reduction also have when a drone transmits a message, in which case the reduction is randomly assigned a number from one to five energy points,

$$RE_{transmit} = e_{drone} - e_{Rand\_Reduction}$$

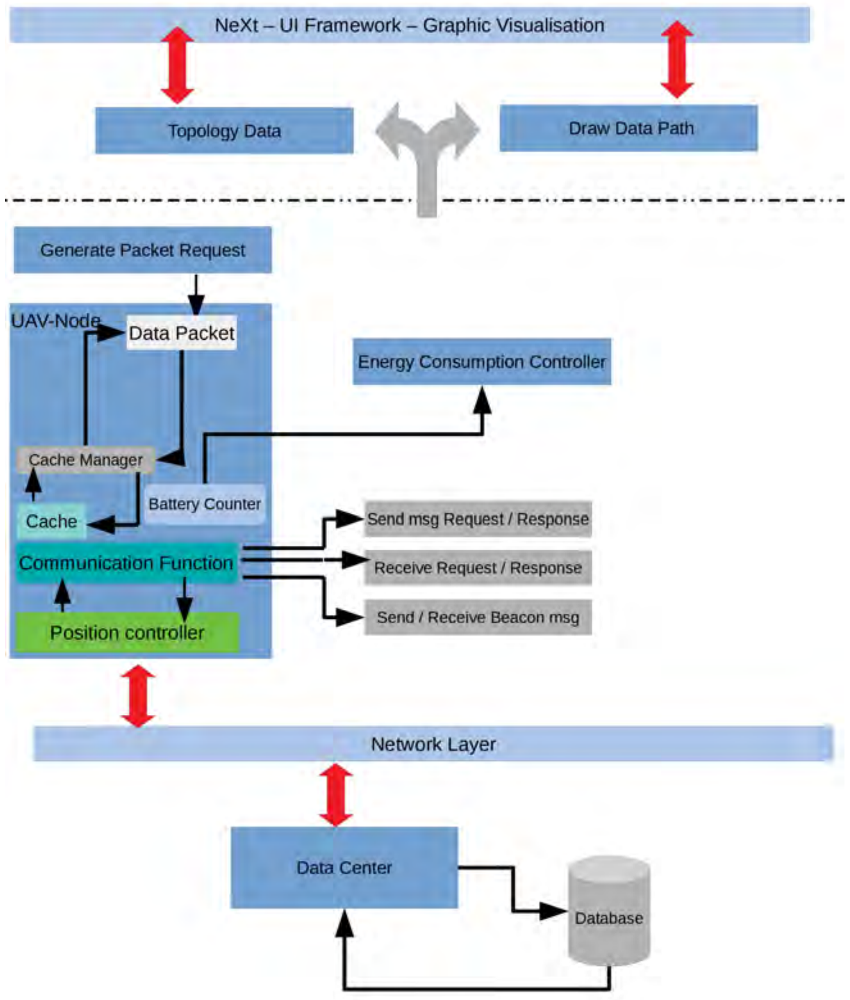


Figure 3.3. Illustration of the overall simulator system

3.2.2 NeXt-UI Framework

The NDR simulator supports graphic user interface, which was build with NeXt-UI framework of Cisco. NeXt-UI is a Javascript/CSS framework for frontend engineers who are in need to embed topology interaction into their web applications. NeXt features MVVP, OOP and DOM manipulation. NeXt renders network topologies and enables user interaction with them through event listeners. We choose to use NeXt UI framework because it provides a network centric topology UI component featuring high performance and rich functionality. NeXt can display large complex network topologies, aggregated network nodes, traffic/path/tunnel/group visualizations and it includes different layout algorithms, map overlays, and preset user friendly interactions. NeXt can work together with DLUX to build ODL apps.

Our back-end simulator defines, calculates and run our algorithm simulations and pass all the data into our local front-end web server in order to visualize our network topology. We build two important files, the Topology_Data and the Draw_Data_Path file as we see in Figure 3.3.

```
        <!DOCTYPE html>
        <html>
        <head>
            <title>Quick start with NeXt</title>
            <!-- NeXt library -->
            <meta http-equiv="refresh" content="4"/>
            <link rel="stylesheet" href="bower_components/next-bower/css/next.min.
    css"     >
           <script type="text/javascript" src="bower_components/next-bower/js/next
    .mi     n.js"></script>
        </head>
        <body >
            <div id="topology-container"></div>
            <!-- Application scripts -->
            <script type="text/javascript" src="app/action-panel.js"></script>
            <script type="text/javascript" src="app/topology_data.js"></script>
            <script type="text/javascript" src="app/topology.js"></script>
            <script type="text/javascript" src="app/main.js"></script>
            <script src="assets/js/jquery.min.js"></script>

        </body>
        </html>

```

Figure 3.4. HTML Server

We include these files in our index.html file as we see the code in Figure 3.4. These files are dynamically generated depending on the number of drones, their movement and their connections. The Topology_Data file is in charge of holding and showing the topology structure of our ad-hoc network. Every drone corresponds to nodes in our file. The form of the Topology_Data has nodes and links as it is shown in Figures 3.5 and 3.6 respectively. The nodes-drones have unique id and name, and their current position is indicated by the values of x, y. Links also have unique id and they are created by declaring the source($drone\_id$) and the target($drone\_id$).

The Draw_Data_Path file draws the drone requests and the drone or Data Center responses by using arrows to demonstrate these actions. An idea of the formation of this file is represented in Figure 3.7. We know the communication links that our network topology have and based on our algorithm we choose to draw our communication arrows whenever it is needed. At the end, we see in Figure **??** an example of the graphic part of our simulator which simulates a flying ad-hoc network.

```
        topologyData = {
            'nodes': [
                {
                    'id': 0,
                    'name': 'Drone_0',
                    'x': 155,
                    'y': 24,
                    'icon': 'dronescan',
            },

```

Figure 3.5. The nodes in the topology

```
        'links': [
            {
                'id': 0,
                'source': 0,
                'target': 1,
            },
        ]

```

Figure 3.6. The links in the topology

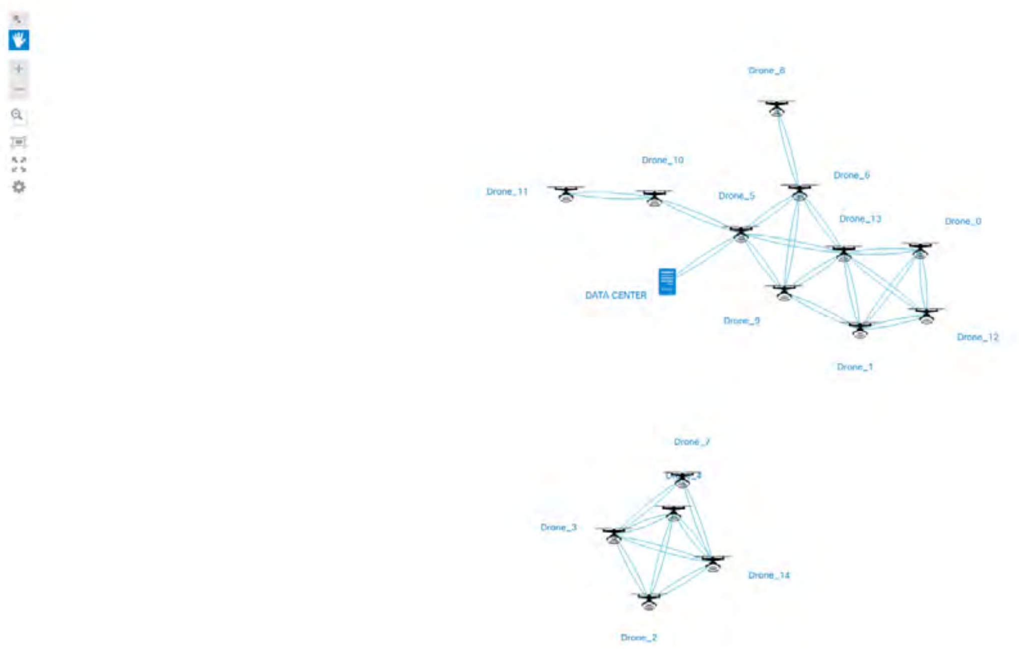
```
        var pathLinks[i] = [
                        topo.getLink(links[i]),
                    ]

            var path[i] = new nx.graphic.Topology.Path({
                'pathWidth': 2,
                'links': pathLinks[i],
                'arrow': 'cap'
            })

```

Figure 3.7. The file that draws the arrows which indicates the communication

(a)



(b)

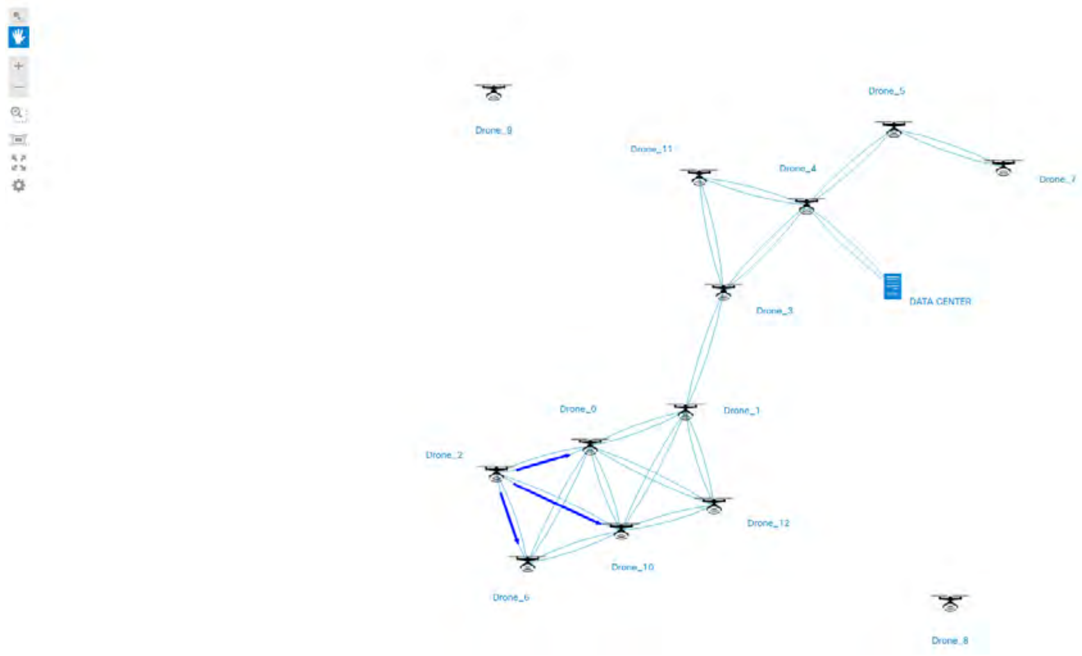


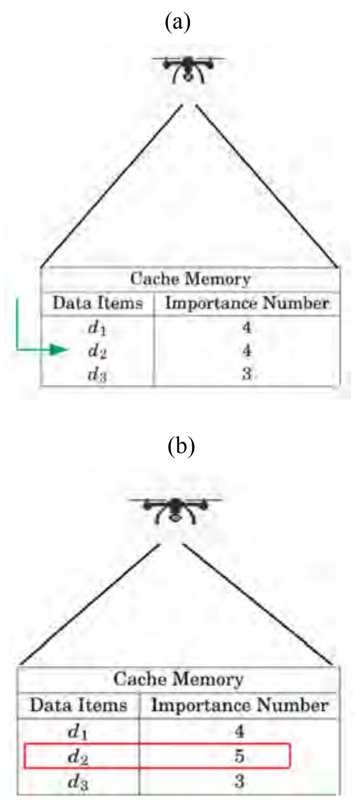
Figure 3.8. (a) A simulation example of an ad-hoc network of drones. (b) Visualize sending one drone messages to its neighbors.

3.3 Energy Efficient Caching

After describing the basis, that is, the system model and the functions of the simulator, we can now proceed to the algorithm (EEC) analysis. The Energy Efficient Caching consists of three phases, the Local Search, Asking Neighbors and the Find Shortest Energy Path to Data Center. We will describe the pieces one by one below.

3.3.1 Local Search

As we analyze above in 3.2, when the drones start to fly the simulator fills their cache memory with data from the Data Center. When a user selects a drone, it calls the function Generate Packet Request, which generates a wish for a data item. So in this phase, once the request for the data item is born, the drone search in its local cache memory to find it. If the operation succeed, we have a local cache hit and it updates its cache. As we mentioned 3.2, every data item in cache, has an importance number which indicates how often this item is accessed or how often it is requested. So every time we have a cache hit we increase by one the importance number, as we see in Figure 3.9. In this case, which is the most optimal, the drone will not send a request to the network. Otherwise, the process moves on Asking Neighbors phase.



(a)

Cache Memory	
Data Items	Importance Number
d_1	4
d_2	4
d_3	3

(b)

Cache Memory	
Data Items	Importance Number
d_1	4
d_2	5
d_3	3

Figure 3.9. (a) Find the element- Cache Hit. (b) Increase Importance Number.

3.3.2 Asking Neighbors

In this phase, if the current drone did not find the data item into its cache, as long as the drone knows its neighbors (from beacon messages), sends a packet request about this data, as we see in Figure 3.8 (b). The neighbor drone that finds the data item into its cache, increase the importance number and send a packet response to the source drone, and in turn saves the data item in cache with importance number equals to one. Although, if the data item found in two or more neighbor drones, then the neighbor drone with the most energy is the one that will respond to source drone request. We can see an instance of the neighbor drone positive response simulation in Figure 3.10 and a in Figure 3.11 a negative neighbor response, which in this case the algorithm moves on the Find Shortest Energy Path part.

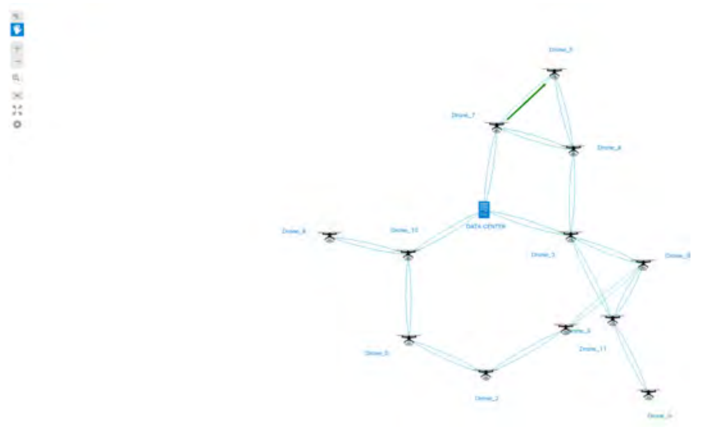


Figure 3.10: Neighbor Cache Hit.

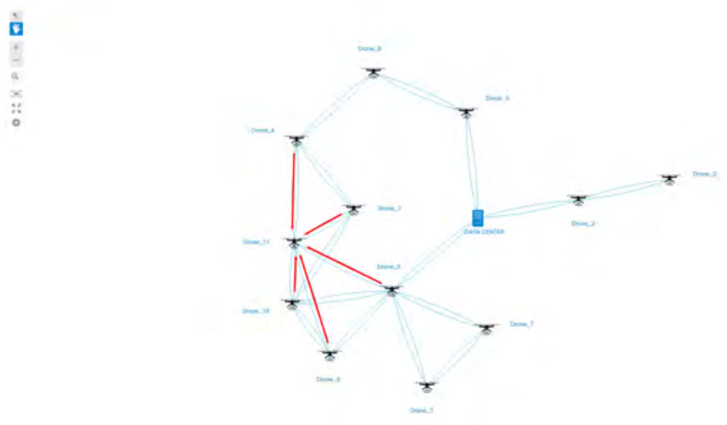


Figure 3.11: Neighbor Cache Miss.

3.3.3 Find Shortest Energy Efficient Path

This is the last part of EEC algorithm. If no neighbor found the data item, they give a negative response and the network undertakes to send the packet request in to the Data Center. The algorithm starts to find the shortest path to Data Center in combination with the remaining energy of each drone.

Every drone start with a maximum energy value e_{max} = 100 and also has a remaining energy value. So we use them in order to assign a weight to each one.

$$weight_i = e_{max} - remaining\_energy_i$$

Immediately after the weight assignment procedure, the algorithm starts to calculate possible tentative distances to the Data Center. A tentative distance from a drone i to a drone j is calculated like above,

$$tentative\_path_{i->j} = \sum_{x=i}^{j} weight_x * wlink(i,j)$$

where $wlink(i,i)$ is a value proportional to the distance between them. We will describe the steps of the algorithm:

1. We keep a list with all the unvisited drones.

2. Assign a tentative distance value to all drone nodes. The initial drone set it equal to its weight and infinity to all other nodes.

3. For the current drone, consider all of its unvisited neighbours and calculate their tentative distances through the current drone. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one.

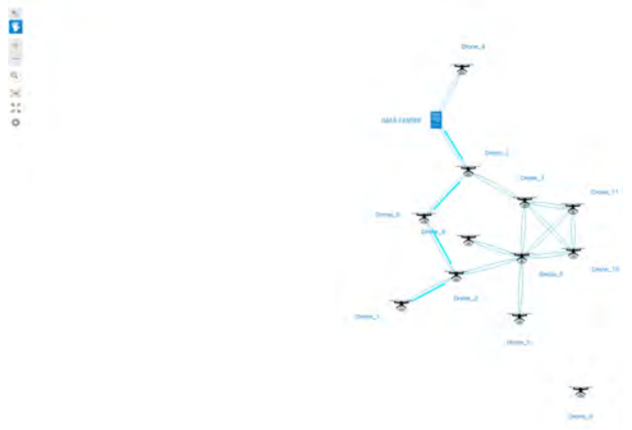
4. When we are done considering all of the unvisited neighbours of the current drone, mark the current drone as visited and remove it from the unvisited set. A visited drone will never be checked again.

5. If the destination drone has been marked, in our case the Data Center, or if the smallest tentative distance among the nodes in the unvisited set is infinity(occurs where there is no connection between the initial drone and the remaining unvisited drones), then stop.

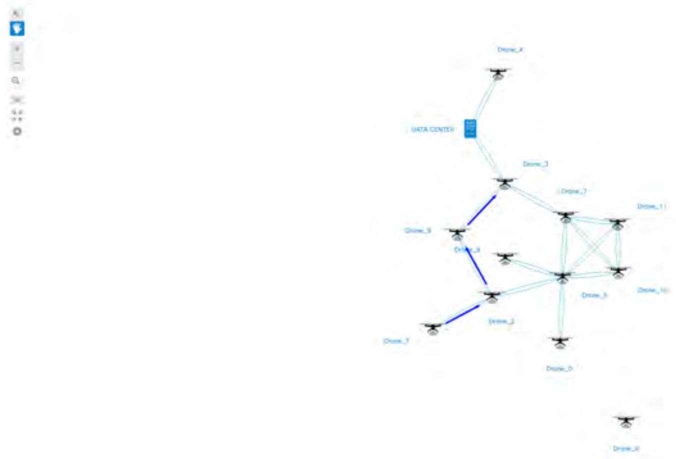
6. Otherwise, select the unvisited drone that is marked with the smallest tentative distance, set it as the new ”current drone”, and go back to step 3.

After finding the shortest energy efficient path, the initial drone sends its data item request through that path. When a non one-hop drone receives a request, it searches its local cache.If it deduces that the request can be satisfied by it (remote cache hit), then stops the request’s route toward the Data Center, and forwards the request back to the source drone. Therefore, during the procedure of forwarding a request toward the Data Center, no searching to other nodes is performed apart from the nodes which reside on the path toward the Data Center. If the data item is not cached by any drone through the energy efficient path, we forward the request until the Data Center and it sends a response back to the source drone. We can see simulator instances of the last part of the EEC algorithm in Figure 3.12 .

(a)



(b)



(c)

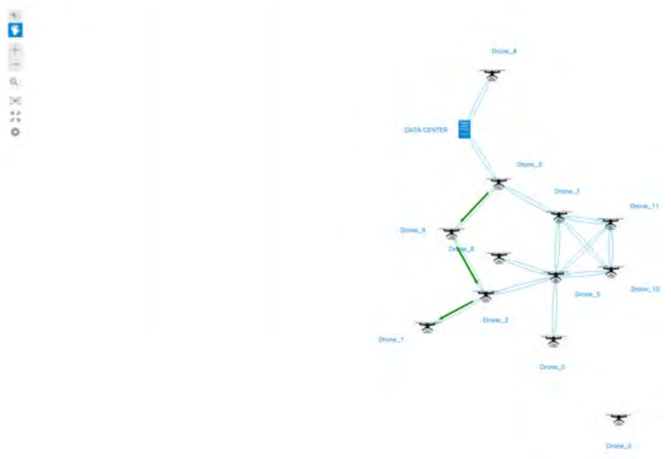


Figure 3.12. (a) Find the shortest energy efficient path. (b) Forwarding the data item request through the shortest energy efficient path. (c) Found the data item before Data Center, Remote Cache Hit

Chapter 4

Experiments Results

Finally comes the stage of actually using the implemented algorithm into our NDR simulator. The first step is to evaluate its cache hits. In order to do this, we will run our simulator applying different number of drones in our ad-hoc network. After this step, a series of executions with different size of cache in order to calculate caching performance with smaller or bigger caches. The final step, simulation experiments will take place in order to calculate the energy consumption of our network.

4.1 Cache Performance

4.1.1 Cache Hits Evaluation

As we describe in section 3.1 our system model, we have a Data Center with all the data items which are necessary for our network. The Data Center may have connections with one or more drones from our network. At the begging we run our simulation with 10 drones, then with 20,30 and at the end with 50 drones. For the first experiment we can see the parameters in Table 4.1.

Simulation Parameters	
Chapters	Value
# data items(N)	50
# drones	10-20-30-50
# Drone Cache Size	32 bytes
# Data Item Size	4 bytes
# Data Center	1
Clock Ticks	50

Table 4.1: Experiment Parameters 1

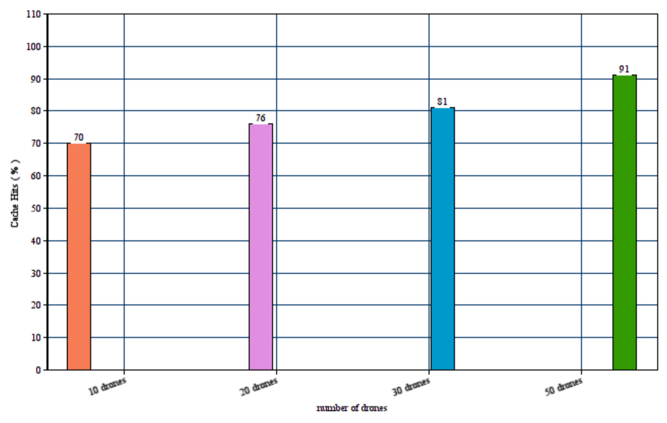


Figure 4.1: Cache hits with different number of drones in our network.

As we can see inf Figure 4.1, for simulation parameters as we defined in Table 4.1, we see that for small caches we have a probability to find our data in the network bigger than 50%. We also notice that for a larger number of drones, we have higher cache hit rates for the same number of data. This means that when we work on larger ad-hoc networks, drones reduce their communication with Data Center as they find the requested data inside the network. Moreover, as we grow the network, we observe that at the begging a drone may have bigger latency ,as we see in Figure 4.2 because the network is still ”cold” and we need to access more drones in order to get our data. Although, as the time pass by, the average latency time may reduced in larger networks as the algorithm tend to store the popular data items over the network.

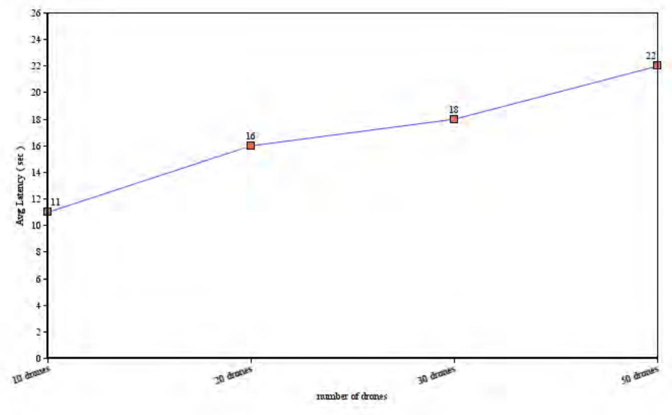


Figure 4.2: Measure the average latency at the begin of the network.

After these results for small caches, we increase the cache size of each drone in order to observe how the network will behave. In this simulation we kept one Data Center and the same number of data items as we see in Table 4.2 and we run it only for 50 number of drones.

Simulation Parameters	
Chapters	Value
# data items(N)	50
# drones	50
# Drone Cache Size	32-64-128 bytes
# Data Item Size	4 bytes
# Data Center	1
Clock Ticks	50

Table 4.2: Experiment Parameters 2 - Increase Cache Size

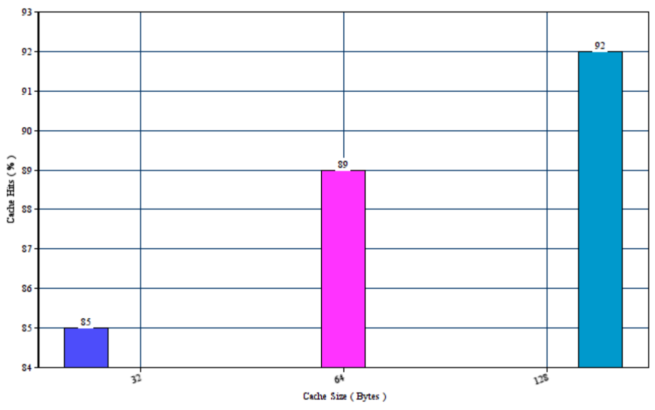


Figure 4.3: Cache hits with bigger caches.

As we see in Figure 4.3, the cache hits rate is increasing when we set bigger caches in drones. This was expected, since we have more storage space for the same amount of data. A drone removes data items less frequent, so we reduce the probability of local cache misses. Also, in a large ad-hoc network, even if a drone did not find the data in its memory, the probability of a neighbor cache hit is bigger because of the size of the network and the size of the cache.

The Figure 4.4 shows that when we use bigger caches in drones we have better latency while we have the same size and number of data items. It is obvious , as we analyze above, that we will have more local cache hits and maybe more neighbor cache hits. Even if we do not find the data in some of our neighbors, it is very possible to find the data while we sending the request through the shortest path to the Data Center.

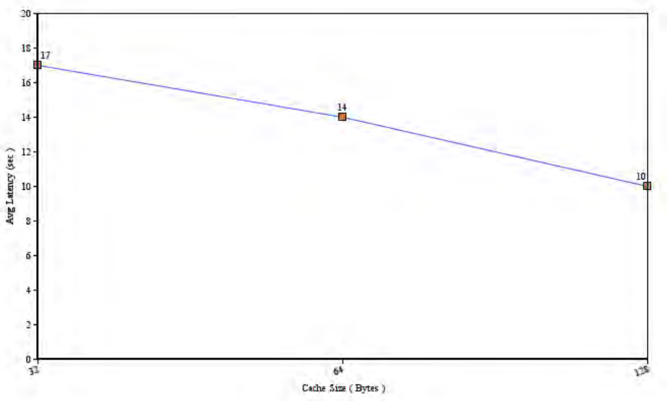


Figure 4.4: Reduce drone latency with bigger caches.

4.1.2 Energy Performance

As we explain in the section 3.3, the Energy Efficient Caching , reduces the energy consumption by using drones that have more remaining energy, because every time we choose to get data from the neighbor with bigger energy or in a neighbor cache miss, we get the data from Data Center through the path with the bigger energy and the fewest hops. To test how our protocol manages the energy consumption we run two type of applications, one with our algorithm and another one where in the Asking Neighbors phase 3.3.2, the chosen neighbor drone is the one who found the data and the distance between the source drone is the smaller one.

Furthermore, the second approach find the shortest path to Data Center without taking into account the energy of the drones as EEC protocol does 3.12. As we mentioned before in section 3.2, the NDR simulator denotes energy units as energy of each drone with max value $e_{max} = 100$. The Table below 4.3 denotes the experiment parameters for our simulation.

Simulation Parameters for energy consumption	
Chapters	Value
# data items(N)	50
# drones	50
# Drone Cache Size	16 bytes
# Data Item Size	4 bytes
# Data Center	1
Clock Ticks	10
Energy Unit	100

Table 4.3: Experiment Parameters 2 - Increase Cache Size

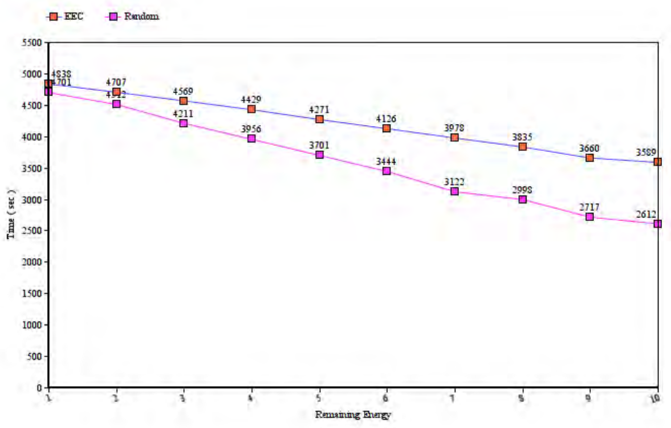


Figure 4.5: Energy Consumption of EEC compare with a random approach.

In the graph 4.5, demonstrates the total energy reduction of the ad-hoc network. Every time unit we calculate the total energy by adding the energy of every drone. So, as we see in 4.5 the EEC protocol seems to treats better the energy of the network compared with the random approach we discussed above. Because we work on a network with a lifetime limit, it is important to manage it intelligently to keep it running as long as possible.

Chapter 5

Conclusion

5.1 Summary

As we mentioned in the introduction the UAV Ad-Hoc networks have many applications and are increasingly used nowadays. But there are still many problems to consider in terms of communication and energy consumption.

The contribution of this dissertation is to provide a distributed energy efficient caching algorithm for flying ad-hoc networks ,the EEC, running to our implemented simulator for drones, the NDR simulator. The algorithm detects which drone can serve the request for a given drone inside the ad-hoc network and also uses, if possible, the drone with the most energy. The algorithm increase the remote hits up to 40% and reduces the energy reduction up to 33%.

5.2 Future Work

The next thing to do is to improve the caching technique by using more drones to find the requested data item before sending a request to Data Center. Another important is to improve the simulator in order to run reliable experiments.

Bibliography

[1] A. I. Alshabtat, L. Dong, J. Li, and F. Yang, *Low latency routing algorithm for unmanned aerial vehicles ad-hoc networks*, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering, (2010).

[2] S. Basagni, M. Mastrogiovanni, A. Panconesi, and C. Petrioli, *Localized protocols for ad hoc clustering and backbone formation: A performance comparison*, IEEE Transactions on Parallel and Distributed Systems, (2006).

[3] I. Bekmezci, I. Sen, and E. Erkalkan, *Flying ad hoc networks (FANET) test bed implementation*, (2015).

[4] O. Bouachir, A. Abrassart, F. Garcia, and N. Larrieu, *A mobility model for uav ad hoc network*, in 2014 international conference on unmanned aircraft systems (ICUAS), 2014.

[5] G. Cao, L. Yin, and C. R. Das, *Cooperative cache-based data access in ad hoc networks*, Computer, (2004).

[6] N. Chand, R. Joshi, and M. Misra, *Cooperative caching strategy in mobile ad hoc networks based on clusters*, Wireless Personal Communications, (2007).

[7] N. Chand, R. C. Joshi, and M. Misra, *A zone co-operation approach for efficient caching in mobile ad hoc networks*, International Journal of Communication Systems, (2006).

[8] N. Dimokas, D. Katsaros, and Y. Manolopoulos, *Cooperative caching in wireless multimedia sensor networks*, Mob. Netw. Appl., (2008).

[9] Y. Du, S. K. Gupta, and G. Varsamopoulos, *Improving on-demand data access efficiency in manets with cooperative caching*, Ad Hoc Networks, (2009).

[10] A. Franchi, C. Secchi, M. Ryll, H. H. Bulthoff, and P. R. Giordano, *Shared control : Balancing autonomy and human assistance with a group of quadrotor uavs*, IEEE Robotics Automation Magazine, (2012).

[11] L. Gupta, R. Jain, and G. Vaszkun, *Survey of important issues in uav communication networks*, IEEE Communications Surveys & Tutorials, (2015).

[12] Z. HAAS, *The zone routing protocol (zrp) for ad hoc networks*, IETF Internet Draft, (1998).

[13] T. Hara, *Cooperative caching by mobile clients in push-based information systems*, in Proceedings of the eleventh international conference on Information and knowledge management, 2002.

[14] S. Hayat, E. Yanmaz, and R. Muzaffar, *Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint*, IEEE Communications Surveys, (2016).

[15] D. B. Johnson and D. A. Maltz, *Dynamic source routing in ad hoc wireless networks*, The Kluwer International Series in Engineering and Computer Science, (1996).

[16] K. Kumaravel, A.Marimuthu, and M. S. M.Phil, *Shortest path exploration in wireless mesh networks improved energy efficiency*, (2014).

[17] J. Li, Y. Zhou, and L. Lamont, *Communication architectures and protocols for networking unmanned aerial vehicles*, (2013).

[18] Liangzhong Yin and Guohong Cao, *Supporting cooperative caching in ad hoc networks*, (2004).

[19] J. G. Manathara, P. Sujit, and R. W. Beard, *Multiple uav coalitions for a search and prosecute mission*, Journal of Intelligent & Robotic Systems, (2011).

[20] C. Phaneendra, K. Srimathi, T. Shekhar, and V. Ravi Kumar, *Caching approach foundation on information density evaluation in wireless ad hoc networks*, International Journal of Advanced Research in Computer Science and Software Engineering, (2012).

[21] H. Shen, S. K. Das, M. Kumar, and Z. Wang, *Cooperative caching with optimal radius in hybrid wireless networks*, in International Conference on Research in Networking, 2004.

[22] H. Xiang and L. Tian, *Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (uav)*, Biosystems engineering, (2011).