Imperial College London

Department of Computing

Drone Network Simulation on SpatialOS
Interim Report

Author:

Paul Balaji

Supervisor:

Prof. William Knottenbelt

Second Marker:

To. Be Decided

January 26, 2018

Contents

1 Introduction

Drone technology is becoming increasingly popular. Their agility and ability to be used remotely makes them ideal for a number of use cases in several industries such as film, law enforcement, emergency services, agriculture and commercial delivery[26].

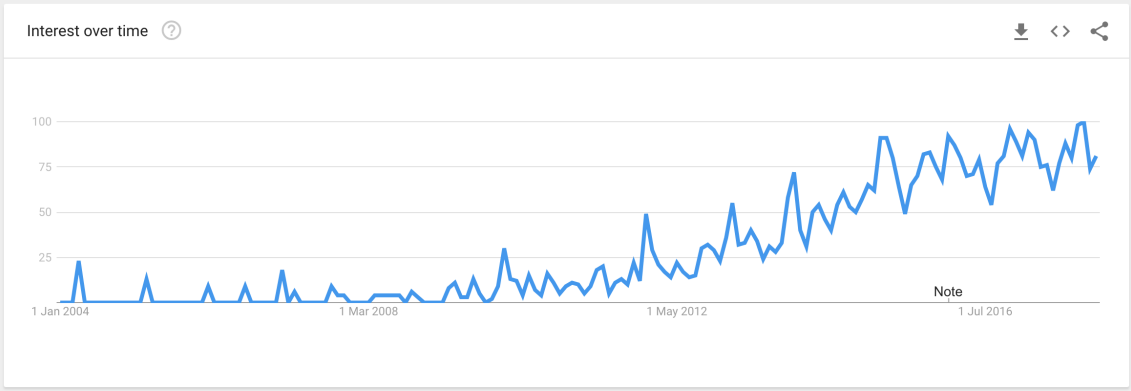


Figure 1: Google Trends for "Drone Technology". [14]

Due to numerous advances in technology, drones are quickly advancing to the point where human input is no longer a necessity. This has led to many companies showing interest in integrating drones with their work in the coming years.

Although it may be an engineer's dream for a fully automated world, drones in particular are a harrowing reminder that there are real risks associated with them. There are already several incidents of drones crashing into planes and flying into areas they shouldn't, most notably near Heathrow airport[5]. All of this provides motivation to introduce some form of autonomous air traffic control system to navigate these drones to their respective destinations in a safe manner.

However, prior work has been done on the routing and navigation aspect of such a system[2]. In order for drones to truly take over more aspects of our lives, we must look at how they can provide a tangible benefit to specific use-cases. There is no doubt that simply removing the human element can save costs drastically, and there is none more exciting an application than with physical delivery networks.

The main factors that could prompt higher adoption of drones for delivery networks are cost, delivery speeds, and convenience. As figure 2 shows, drones can be both faster and cheaper than existing deliver options that consumers have access to.

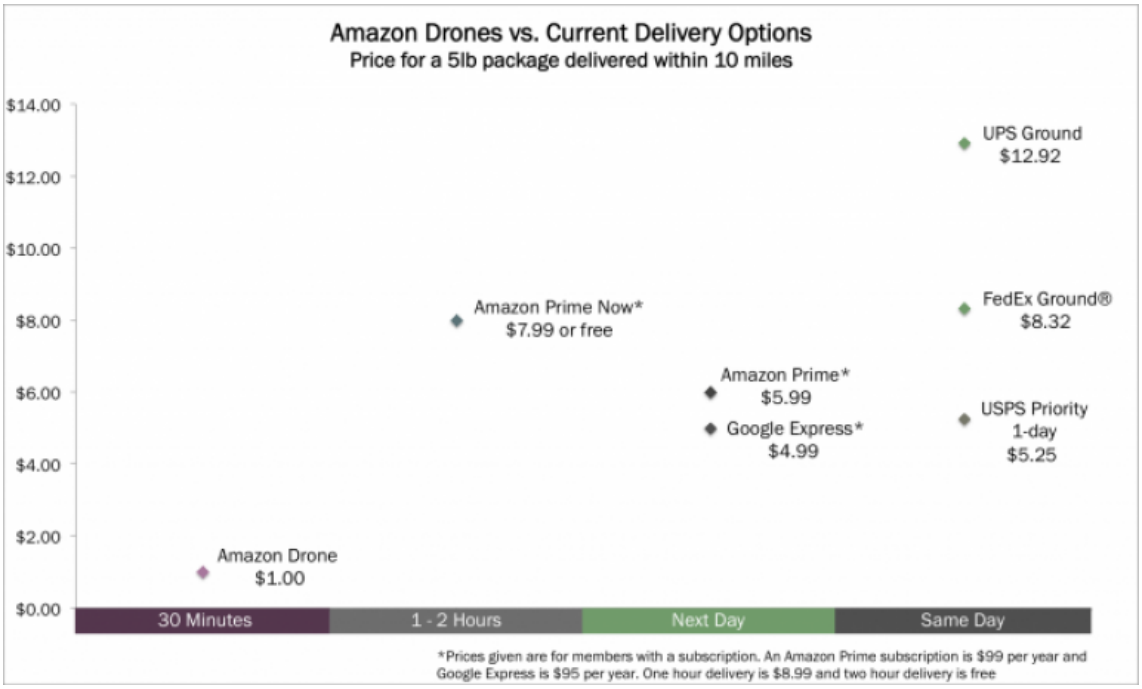


Figure 2: How Amazon Air fits into the wider delivery market. [41]

It appears then that Amazon stands to increase their margins considerably if they can successfully pull off their Prime Air initiative[42], but to do this they need to be able to handle the additional problems of scheduling and load balancing massive quantities of drones, on top of the existing routing problems.

In a more general sense, goods delivery networks have a lot to gain by optimising for profit, and to do this better we consider a variable delivery pricing mechanism that factors in the quality of service provided to the user. A high-priority product may warrant a higher delivery fee for quicker delivery but an similarly high penalty for being late. At the same time, a low-priority product may not have any such penalty, instead opting for a low delivery fee. Being able to factor this new fee model when scheduling tasks and allocating drones may be the next step in the ongoing automation of last-mile delivery[24].

Regardless of how such a model is implemented, it needs to be tested thoroughly before being deployed into production with actual drones. Considering a typical delivery drone could cost around $100 to $500[29], and that there is likely be a lot of failure before a working implementation, there could be several hundreds of thousand dollars worth of losses.

Therefore, our best port of call is to simulate scenarios that are as close to the real world as possible. One tech firm investigating how to produce richer, meaningful and more realistic simulations is *Improbable*[18] - a London-based tech startup actively developing a distributed simulation platform called *SpatialOS*[20].

We aim to leverage the power of SpatialOS to produce novel simulations of a drone delivery network. The simulation will build upon existing routing solutions[2] and would aim to schedule deliveries based on a Quality of Service value curve. Furthermore, we wish to show how the net profit of a drone delivery network with this form of scheduling might compare to delivery networks that utilise a different scheduling mechanism.

2 Background

We first provide an insight into new developments in the physical delivery network sector, and then specifically considerations to account for when integrating drone technology in our day to day lives. Additionally we summarise prior work completed by Imperial students on an autonomous air traffic control system for drones.

We then continue to discuss methods of prioritising a delivery network for financial gain. Finally, we give details about Improbable's SpatialOS and the reasoning for using this platform for the drone simulation.

2.1 Future of Delivery Networks

From the ancient days of Assyrian trade in 19 BCE [37], to the modern day online market - the ability to trade goods and services has had a profound impact on the way we live our lives. Thanks to globalisation and the explosion of the internet, we are now able to use a service such as Amazon Prime and get goods delivered within days, if not hours, of purchasing through a simple click. Through it all, physical delivery networks enable this online retail behemoth.

As technology evolves, so too will these delivery networks - and many companies are currently looking into how to leverage upcoming tech[28] in order to generate hype for their business and, more importantly, increase their profit margins. Some promising developments are delivery robots, autonomous cars and, of course, drones.

2.1.1 Delivery Robots

Delivery robots are unmanned devices that are designed to be operated on pavements and cycle lanes at low speeds of about 4 miles per hour[28]. Although the unmanned device market is in its infancy, there are already prototypes hitting the market such as Starship's robot[27], which can hold upto 20-25 pounds, and Dispatch's "Carry"[25], that holds up to 100 pounds.

Figure 3: Starship's robot confronting a senior citizen. Image from Engadget[36].

Due to their on-the-ground nature, delivery robots are primarily designed for low-footfall suburban neighbourhoods, closed residential areas or campuses. They would be scheduled through an Uber-like application that may allow users to track the location of the robot, for the customer's knowledge, and to also unlock the robot once it reaches its destination[16].

Thanks to their similarities to autonomous cars, they may have regulatory advantages compared to drones because much of the legislation coming through for autonomous vehicles can apply to delivery robots as well. However, there are profitability concerns because each robot can cost hundres of dollars and there is at this stage there is a high risk of failure, damage and even theft.

Despite some drawbacks, it is evident these robots are designed for short journeys in the so-called "first-mile delivery" sector - with exciting use cases being the delivery of flowers, groceries and possibly medication.

2.1.2 Autonomous Vehicles

Once the de-facto icon of science fictional futurism, autonomous cars are now a close reality. Companies such as Tesla are doing pioneering work in bringing self-driving cars[38] to the general consumer. Meanwhile, other businesses like Uber are experimenting with autonmous taxi services[13].

In fact, Google has obtained a patent for an autonomous delivery truck [15] with different compartments that could be unlocked by validated package recipients - much akin to an Amazon Locker-on-wheels. If not this, these vehicles could be used as an autonomous courier to pick-up locations that users could visit to retrieve their packages.

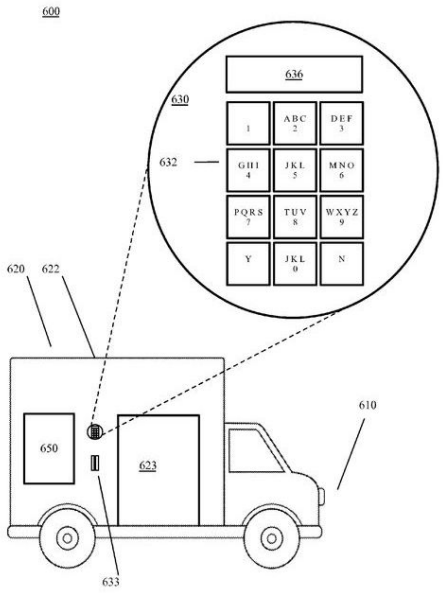


Figure 4: Illustration of Google's Delivery Truck. [31]

Although current regulations dictate that a driver must remain present in the vehicle in the event of software performing differently to expections, this is a necessary precaution whilst driverless systems are still being tested - with further amendments to these laws in sight once these systems can be deemed safe for public use[7].

2.1.3 Drones

The concept of drones has been around for decades, particularly in the military, but only in recent times has the technology evolved to a point where anyone can walk into a consumer electronics store and purchase a drone for their own private use. Though these are manned, it is admirable that many witness the benefits of drones in film/television[40], agriculture[23], policing[4] and other exotic activities[34].

Being able to fly allows drones to not be held hostage to traffic jams or congestion - after all, the sky is a far greater expanse than even the most intricate of road networks. Not only that, they can literally travel "as the crow flies" - allowing them to access remote and previously hard-to-reach areas.

Despite several near-misses [9] and accidents[3] at just Heathrow airpot alone, engineering teams across the world are working on ways to improve drone visibility to its surroundings - one such technology being automatic dependent surveillance-broadcast (ADS-B) [10]. Drones with ADS-B work by determining their own geolocation and velocity in order to broadcast this back to the global network.

ADS-B is already widely used in aircraft, enabling flight-tracking services such as flightradar24 to exist[11], but recent developments have allowed the ADS-B units to be compact and cost-effective enough to be suitable for drones. As more drones adopt this technology, instead of having to get instant updates from a drone directly, autonomous drone traffic control systems could make use of the shared global network for their scheduling and routing calculations.

Over time, an increasing percentage of the world's drones will likely join these shared autonomous systems. This could enable different delivery networks to manage their own fleets as their own air traffic controllers, all the while making use of the shared pool of readable data to identify, intercept and avoid drone collisions at the earliest stage in their pathfinding as possible.

2.2 Drone Considerations

Drones stand to be a revolutionary part of our lives as we welcome the new, incoming era of automation. However, to be practical there are a few key concepts one must understand to ensure that they remain a help and not a hinderence to mankind.

2.2.1 No Fly Zones

No Fly Zones (which we will abbreviate as NFZs) are geographical areas where a drone is not allowed to enter or fly at any altitude. Examples of these may include Hyde Park, Buckingham Palace, airports and military locations. Typically these are static obstacles that will always remain a NFZ, however we could also consider cases when they could be created dynamically.

Suppose there was an issue of national security, it would be beneficial for the security and intelligence services to set up a temporary NFZ around areas where they deem a risk so that they can conduct their own operations without worry of external parties interfering with the situation.

2.2.2 Manned Aviation

Manned Aviation is any form of airborne transport with humans on board. This would include passenger jets, private jets, helicopters and other miscallaneous vechicles. For simulation purposes, we can consider these to have straight paths from a start to an end because relative to the zig-zagging of drones - they are effectively straight.

It is paramount to avoid collisions with manned aviation because there is a high risk of human calamity, in addition to the bad press and financial costs associated with such an air traffic incident.

2.2.3 Other Drones

Naturally we would want to make sure that we avoid colliding into other drones as well. In a perfect system, a single air traffic controller would have totalitarian dominance over all drones that take to the skies - however this is not *Black Mirror* and we have to assume that there will always be drones that this system will not be able to control or predict.

Nonetheless, a system can ensure that it navigates drones under its control as best as it can, avoiding these external drones and other rogue drones that may be flying with the sole intention of causing problems to others.

2.2.4 Toll Zones

Toll Zones are geographical areas where a drone has to pay an extra fee to pass through at any altitude. It is a very similar idea to the one that led to Congestion Charges being applied to much of central London. By restricting certain areas only to those who are willing to pay the fee to use the airspace, it reduces the density of air traffic in a specific area. These charges could also be used as an incentive to reduce pollution in the toll zone, and the additional revenue generated by the toll fees could be used towards the drone-related systems in place within the zone.

2.3 Autonomous Air Traffic Control (AATC)

2.3.1 What is AATC?

During the Autumn Term of the 2016-17 academic year, several students undertook a group project in association with Microsoft and Altitude Angel to produce an autonomous air traffic control system. The goal of the project was to safely navigate drones from their start to their end goals, whilst avoiding obstacles and taking the shortest possible path to minimise battery use.

2.3.2 Technical Overview

AATC has a simple client-server architecture, where drones connect to the REST service to send their telemetry information and goal waypoints every second. The server sends back direction recommendations to navigate the drone to its destination.

Because it would have been too expensive to trial the system on actual drones, a test bench was also implemented that would simulate the drones polling the server and responding to its recommendations. This test bench produced a set of simulation data, that is then available to view on the AATC visualiser[1].

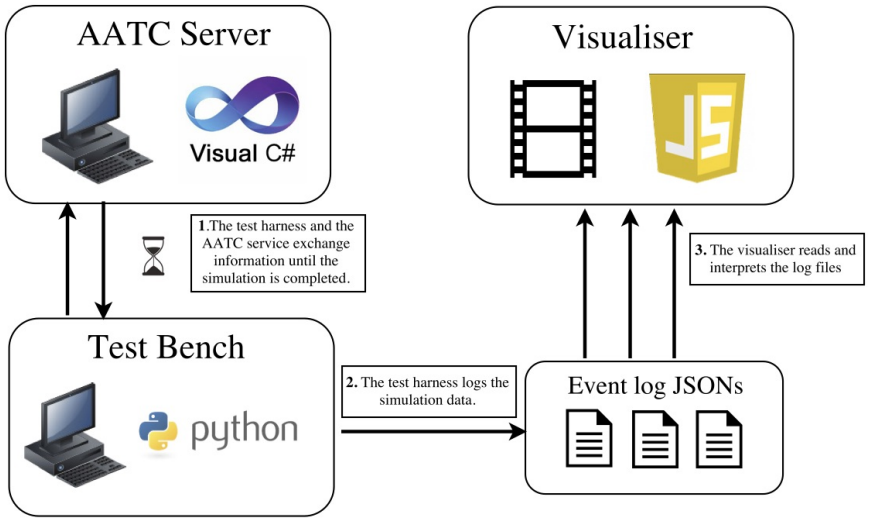


Figure 5: Technical Overview of AATC. [2]

The system was designed with three challenges in mind. The first challenge was to route drones from their origins to their destinations, and secondly, to ensure they avoided collisions with both static and dynamic obstacles - such as the ones mentioned in Section 2.1. The last challenge is to design the system in such a way that it would be able to scale to hundreds and thousands of drones.

To this end, a *Global Layer* was built to deal with static obstacles (such as NFZs) by calculating a path for the drone around NFZs before it begins its journey. The second problem was tackled with a *Reactive Layer* that handles non-static obstacles

(such as manned aviation and other drones). This is the layer that would be providing the real-time updates to drones as they poll the server every second.

Only one drone type was used in AATC, with its specification outlined in Figure 6. It has a radius of 1m and a maximum velocity in any direction of 5 metres per second.

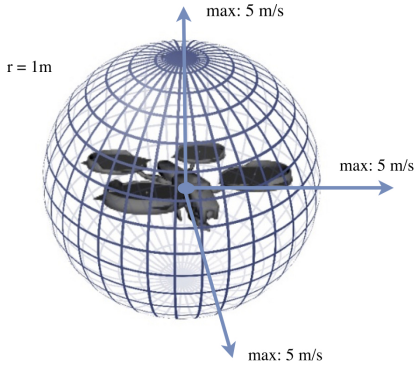


Figure 6: The default AATC drone model. [2]

2.4 The Global Layer

The Global Layer holds the static representation of the world so that given a start and end, it can compute an optimal set of waypoints that a drone should follow - thereby dealing with AATC's path-finding problem.

2.4.1 Dijkstra's Algorithm

Dijkstra's algorithm is a popular algorithm to find the shortest paths between nodes in a graph. When using a co-ordinate grid system, each coordinate could represent a node in a graph and thus the algorithm can also be used to find the shortest path between a source and destination.

See Appendix A for an example python implementation of this algorithm.

However, in the real world, we also have to consider the cost of computation and potentially make use of heuristics in order to return the shortest path given a limited amount of time. Especially in the drone case, we want to compute a good path as quick as possible in order to let the drone proceed along its way.

2.4.2 A* Algorithm

Enter, the A* algorithm. This algorithm is a generalisation of Dijkstra's algorithm that reduces the number of nodes explored during the search process by use of a heuristic - typically a minimum distance to the destination. A benefit over Dijkstra in particular is that it considers the distance already traveled into account, which aids the heuristic mechanism.

Naturally, by searching less nodes on a graph, less computation is performed and therefore A* has better performance than just using a pure form of Dijkstra's algorithm.

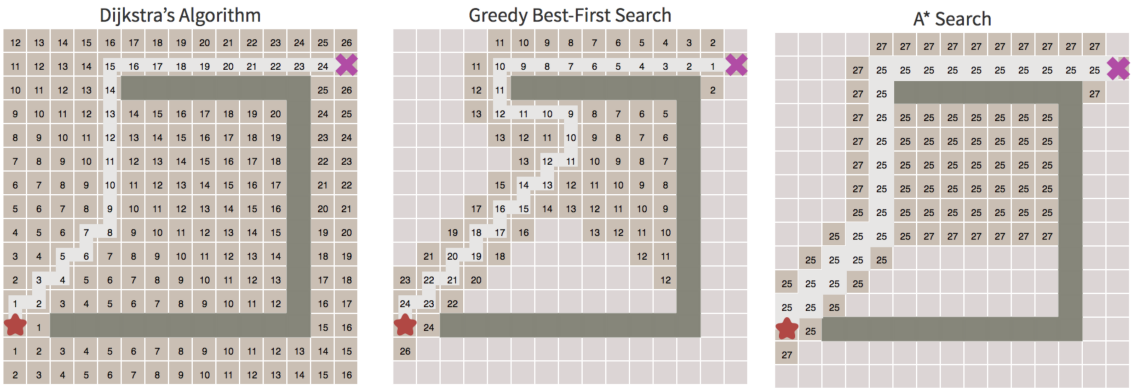


Figure 7: Comparing popular search algorithms. [2]

But for all these positive aspects, one must remember that our use for this algorithm is to compute paths in a real world, which is not necessarily split up into a nice, clean grid. So we turn our attention to a modification of the A* algorithm: Theta*.

2.4.3 Theta* Algorithm

The Theta* algorithm is an any-angle pathfinding algorithm based on the A* algorithm that introduces Line of Sight (LOS) checks, which means that each jump from node to node in the returned path can be at any angle and not just up, down, left or right. This neat addition allows Theta* to be capable of finding near-optimal paths with a runtime comparable to A* [39].

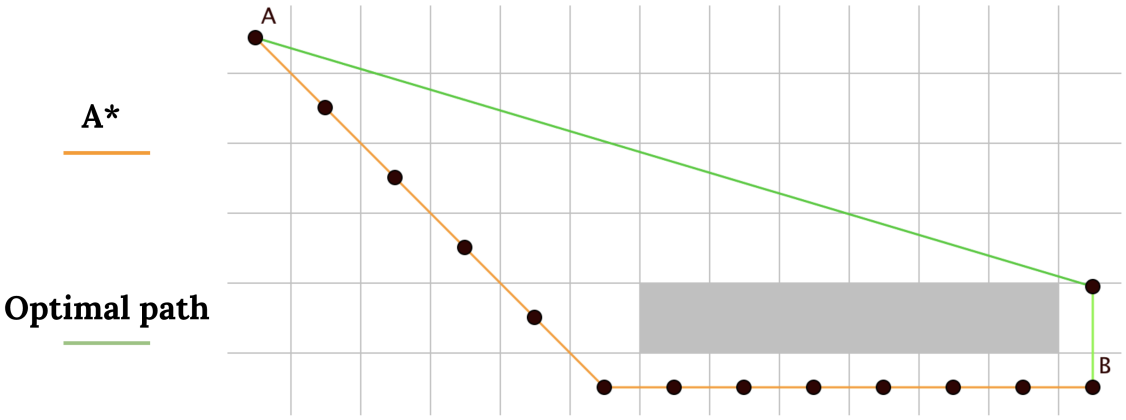


Figure 8: A* vs Optimal Path. [2]

Being able to get as short a path as possible is absolutely vital in the drone use-case because they have limited flying time. After all, their batteries can only last so long before they need to be recharged. Therefore, it is important to ensure drones travel as little distance as possible in order to maximise the number of times they can be used between charges.

2.4.4 Lazy Theta* Algorithm

A further optimisation is to reduce the number of LOS checks performed, as found in another paper by the original authors of the Theta* algorithm[33]. This algorithm assumes every node is in line of sight of its parent, and the LOS check is only performed once the child node is expanded on. If this turns out to be false, then the algorithm defaults to a typical A* approach. Reducing the number of LOS checks therefore improves the algorithm's overall performance.

2.4.5 Conclusions

As seen on Table 2, A* has far better execution times than both Theta* variants but when we get to Table 1 it is evident that Lazy/Theta* produces better paths because the total distance travelled by drones is fewer. Therefore, Lazy Theta* was chosen as the core algorithm for the Global Layer.

Test Case	A*	Theta* and Lazy Theta Star
Queen Threat	6.11	5.98
The Imperial Tunnel	10.83	10.22
The Great London Beehive	52.63	49.77
The Nightmare of Hyde Park	118.4	115.32
The Great Wall of Imperial College	16.37	15.46
Multi Drone collision	15.27	14.82

Table 1: Total path distances for all drones (in km). [2]

Test Case	A*	Theta*	Lazy Theta*
Imperial Tunnel	118	141	124
	120	138	132
	121	137	129
Imperial Tunnel Mean	119.7	138.7	128.3
The Great London Beehive (GLB)	859	1318	1166
	916	1486	1075
	894	1555	1201
The GLB Mean	889.7	1453	1147.4
The Nightmare of Hyde Park (NHP)	9771	28436	21330
	9687	27970	24786
	10118	28273	21609
The NHP Mean	9858.7	28226.4	22575
The Great Wall Of Imperial College (GWIC)	394	913	838
	447	910	840
	416	912	841
The GWIC Mean	419	911.7	839.7

Table 2: Execution times for pathfinding algorithms on AATC test cases. [2]

2.5 The Reactive Layer

Given the list of waypoints that the Global Layer generates, the Reactive Layer's task is to provide the right speed and direction to drones whilst taking into consideration any dynamic obstacles that the drone may face, such as manned aviation, other drones and potentially dynamic NFZs.

2.5.1 Artificial Potential Fields (APF)

A novel way to approach this layer is to create an Artificial Potential Field (APF) in the area the drone operates in, and give each point in this field a potential[43]. By having a low potential for the destination and large potentials at obstacles, the drone simply has to move in a way to get to point of lowest potential. It is analogous to a magnet in a magnetic field, repelled by obstacles and attracted to its destination.

Although one might argue that the Global Layer is not needed anymore because the Reactive Layer solves the pathfinding problem, there are a few issues that arise with this idea. As Figure 10 shows, a U-shaped object poses a problem because the drone may be attracted to the destination and then quickly repelled by the obstacle, and then back to being attracted - and this cycle is seemingly endless.

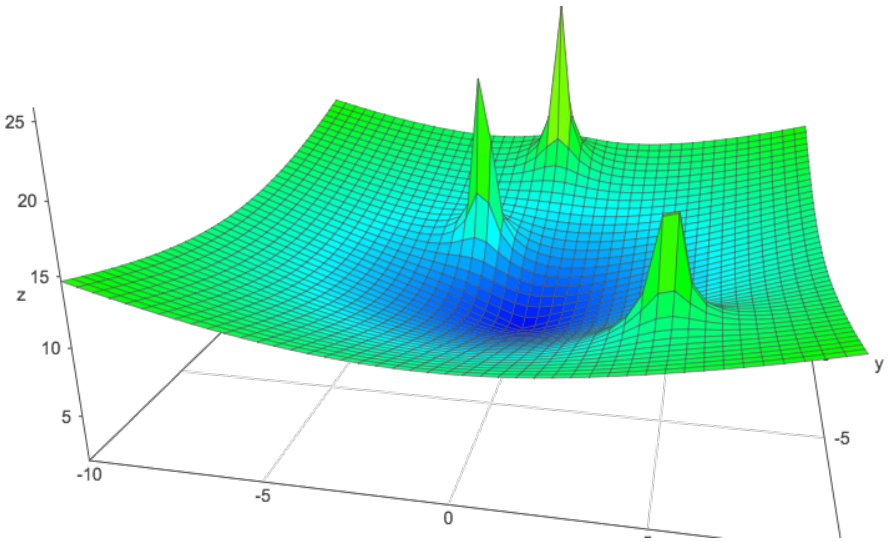


Figure 9: APF with obstacles (green peaks) and a destination (blue trough). [2]

By introducing a random element, as in Rotating APF, the object is repelled in a slightly different direction each time to make it out of the trap and eventually reach its destination. Even this method, however, does not actually guarantee that the drone makes it past the U-shaped obstacle because it depends on how the random element behaves and also whether the drone has enough battery to be loitering for long.

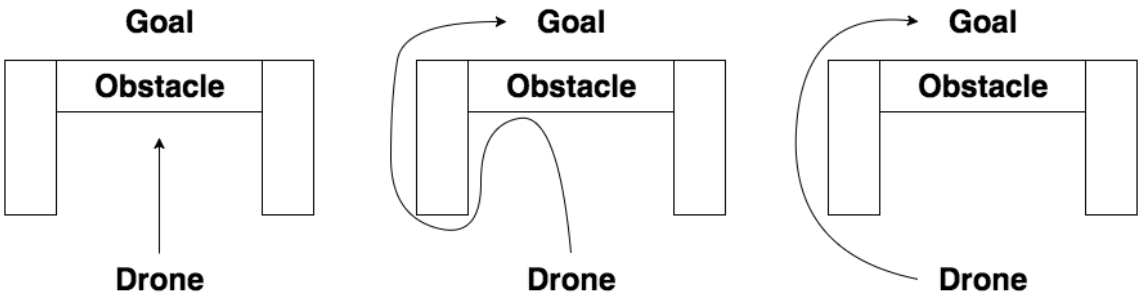


Figure 10: Pure APF vs Rotating APF vs AATC Implementation. [2]

Clearly, integrating both layers proves most fruitful - as the waypoints generated by the Global Layer are used to quite literally navigate around the problematic aspects of the Reactive Layer. This integration that was implemented in the AATC project.

2.5.2 Equations for APF

The full set of equations used to obtain the recommended unnormalized velocity v_{raw} of the drone is:

$$U_a = \rho_a d_{goal}$$

$$U_r = \begin{cases} \rho_r \frac{1}{d_{obst} - d_{safe}} & d_{obst} \leq d_{influence} \\ 0 & otherwise \end{cases}$$

$$U_{ret} = \rho_{ret} d_{last}$$

$$U = U_a + U_r + U_{ret}$$

$$v_{raw} = \nabla U$$

where ∇U is the gradient of U, and U is the magnitude of potential at a point[2].

Since ∇U is the recommended velocity, it can be computed from first principles:

$$\nabla U(x,y,z) = \begin{pmatrix} \frac{\delta U(x,y,z)}{\delta x} \\ \frac{\delta U(x,y,z)}{\delta y} \\ \frac{\delta U(x,y,z)}{\delta z} \end{pmatrix} \approx \begin{pmatrix} U(x+1,y,z) - U(x,y,z) \\ U(x,y+1,z) - U(x,y,z) \\ U(x,y,z+1) - U(x,y,z) \end{pmatrix}$$

The recommended velocity can now be computed by first calculating the potential at 4 points. To calculate the potential at a given point, one must find the sum of:

- U_a - attraction potential using euclidean distance to next waypoint.
- U_r - repulsion potential using euclidean distance to nearest obstacle.
- U_{ret} - return potential using euclidean distance to point from last time step.

To allow drones to gently come to their goals instead of shooting past and potentially missing, the recommended speed is calculated by taking the minimum of the drone's max speed and the distance to the goal:

$$speed = min(max\_speed, d_{goal})$$

The gradient ∇U is then normalised to be a unit vector before being multiplied by the speed to compute the final velocity.

2.5.3 Genetic Algorithm

By paying close attention to Section 2.5.2, we can identify several undefined constants: ρ_r, ρ_a, ρ_{ret} and $d_{influence}$. The balance between them is key, because if ρ_a is too high relative to ρ_r then the drones ignore obstacles and die on impact, but if ρ_r is too high then they will oscillate between objects and never reach their goal. After much fiddling about with "magic numbers", a genetic algorithm was employed to provide the optimal values for the specific drone model.

Genetic algorithms are a metaheuristic inspired by the process of natural selection, and they are a way by which high-quality solutions can be generated for optimisation problems[30]. In the case of AATC, the aforementioned constants need to

be fine-tuned in order to produce an APF model that, to its best ability, does not return absurd, erroneous velocities. See Figure 11 as an example.

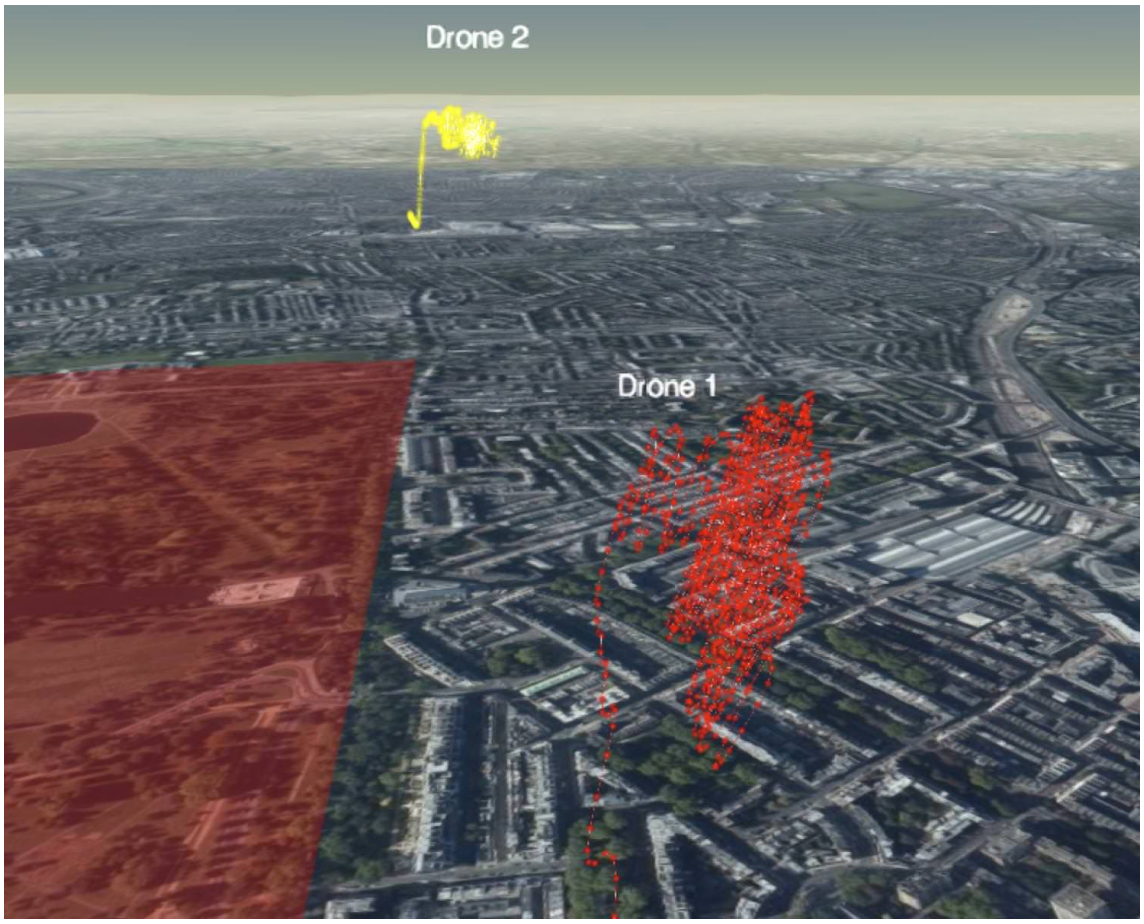


Figure 11: Drone paths after increasing repulsion constant by 3 orders of magnitude.

Effectively, an initial range of values (see Table 3) is provided and the genetic algorithm cycles through all of these, letting these constants compete against each other to see which set of constants produces the simulation with the lowest cost. This cost is determined from a cost function that returns the remaining battery life of a drone if it reaches its destination, or a relatively huge number otherwise.

$$cost = \sum_{testcase} \sum_{drone} \begin{cases} batteryUsed & if\ drone\ reached\ destination \\ 100000 & otherwise \end{cases}$$

Constant	Min	Max	Step	Total
Attraction	0.8	1.1	0.1	3
Repulsion	100	500	100	5
Return	0.1	0.7	0.2	4
Influence distance	300	600	100	4
Population				240

Table 3: Initial configuration of the genetic algorithm for AATC. [2]

2.5.4 Conclusions

In the end, the genetic algorithms only improved the simulations by about 2%[2] which could mean that the original range of constants was already a good set to work with. However, there is still scope for even greater improvements if the genetic algorithm was to be run with a greater number of test scenarios, bigger range of values to cycle through, and generally larger simulations.

2.6 Scalable Drone Network Architecture

The *Internet of Drones* is a "layered network control architecture" to deal with the growing presence of drones in a secure, scalable manner[12]. The author notes that as the volume of drones hits several thousands for a limited region, any centralised air traffic control system would not be able to handle the load, thus leading to the contribution of a decentralised solution.

Drawing inspiration from other important network models such as Air Traffic Control, Cellular networks and the Internet - the author also goes on to state that separating the problem into different layers helps to abstract scalability issues, improve maintainability of the codebase and allow greater flexiblity to improve specific layers as time goes on[12]. Therefore, the existing AATC model might be considered a good candidate to implement this scaling solution on top of.

The author considers splitting the problem into several layers:

1. **Nodes** - points of interest connected together by *airways* and *intersections*.

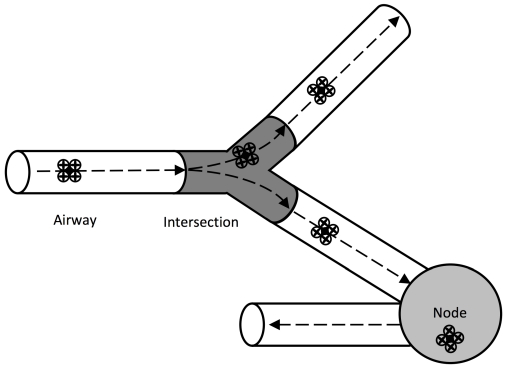


Figure 12: Nodes, airways and intersections.

2. **Zones** - collections of nodes, airways and intersections within a given area.

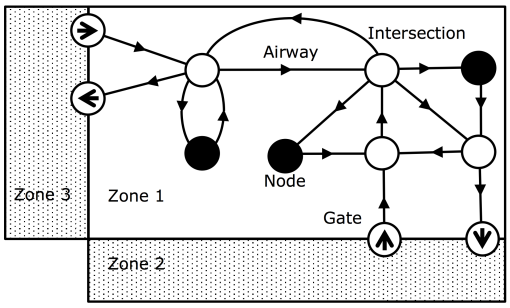


Figure 13: A zone graph, with arrows indicating airways.

3. **Zone Collections** - how zones interconnect through *gates*.

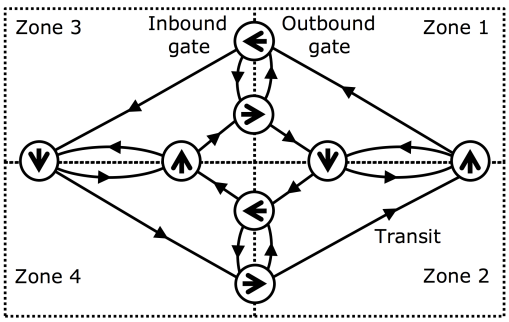


Figure 14: Inter-zone graph, note there can be more than 2 gates at zone boundaries.

2.7 Scheduling Algorithms

In this section we look at a few well known CPU scheduling algorithms[6]. Although CPU scheduling is not part of this project, being able to understand a few different approaches to the problem will give insight into how to schedule drones later on. However, there are two key difference between drone and CPU scheduling.

The first is that CPUs can time slice different tasks whereas once a drone is scheduled onto a task, it has to complete the whole task before being able to pick up a new one. Secondly, there are vastly more drones that could be running at a point in time compared to the number of cores in multi-core CPU, thus there is massive parallelism.

2.7.1 First Come First Serve (FCFS)

As the title says, this algorithm operates as a FIFO queue where tasks are scheduled in the same order that they arrive. Some might consider this the fairest form of scheduling, since it does not discriminate against any task, but simply prioritises the one that arrived soonest.

2.7.2 Shortest Job First (SJF)

This approach looks at the time each incoming task would take to complete, and puts the shortest task at the front of the schedule. While this means that small tasks get completed quickly, a constant stream of small tasks could mean that any heavier tasks never get scheduled or completed.

2.7.3 Priority Scheduling

Priority scheduling is a more general case of SJF, since each task is now given a priority - defined either internally or externally - and tasks with the highest priority are scheduled first. In SJF, the inverse of the estimated job time is used as the priority.

2.7.4 Multi-Level Feedback Queue (MLFQ)

MLFQ is an extension of an ordinary multi-level queue, where there are several queues of tasks to be completed and tasks are scheduled from non-empty queues in a round-robin fashion. A benefit of this approach is that each queue could be running different scheduling algorithms that better fits the priority of that queue.

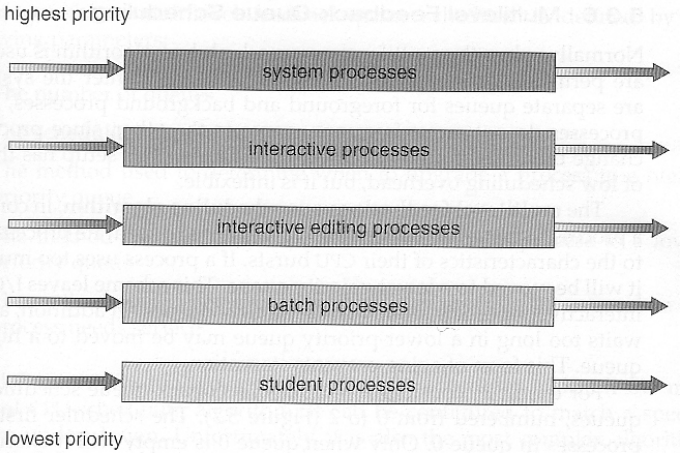


Figure 15: Multiple levels of queues, scheduled in a round-robin fashion. [6]

The "feedback" element of MLFQ allows tasks to be moved from one queue to another, depending on changes of circumstances. For example, if a queue has been left in a low priority queue for long enough, it may be moved to a higher priority queue - thus improving the chance that all incoming tasks get scheduled at some point in time.

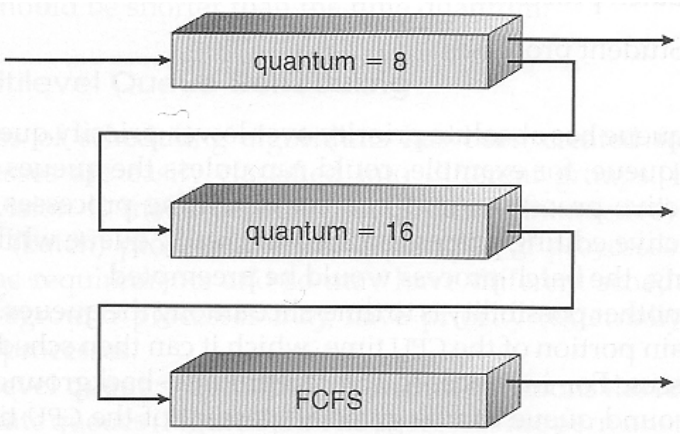


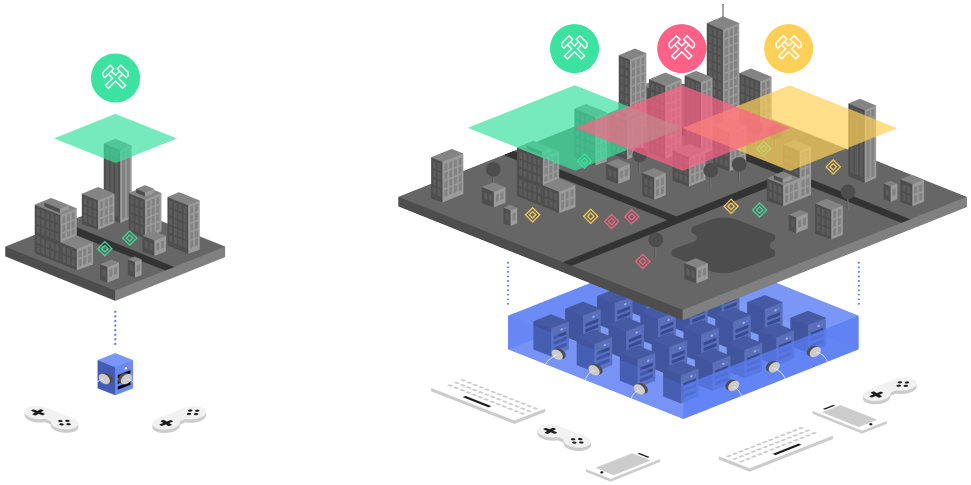
Figure 16: Arrows indicate tasks can be moved up one queue to another. [6]

2.8 SpatialOS

SpatialOS is a platform, created by *Improbable*[18], for running massive-scale simulated worlds. In their own words[20]:

> SpatialOS is a cloud-based computational platform that lets you use many servers and engines to power a single world. The platform coordinates a swarm of micro-services called workers, which overlap and dynamically reorganize to power a huge, seamless world. The platform also lets you handle a huge number of concurrent players across different devices in one world.

Although most commonly used for large distributed game worlds, SpatialOS has been used for simulating cities[32], the backbone of the Internet[22] and a model of the brain[21]. It is an ideal platform for simulating "independent entities that have a location in space"[21].



(a) The current approach. (b) How SpatialOS is different.

Figure 17: Comparing approaches to multiplayer. [20]

Anybody is able to sign up and use SpatialOS to build games using either the Unity3D or Unreal game engines. Alternatively, one can use the C, C++ and Java SDKs to create custom workers to be ran on SpatialOS deployments.

A benefit of SpatialOS's Unity and Unreal SDKs is that creating a sole game client, without manually writing network code, is almost enough to get multiplayer built-in to the game for "free", i.e. taken as granted.

2.8.1 Unity SDK

Of the two game engine integrations, the Unity SDK is arguably the most stable SDK because it has been in development for far longer than the Unreal SDK, and it is not in beta. From prior industrial experience, the Unity integration is also easier to implement quick prototypes with and iterate on.

2.8.2 Abstraction

SpatialOS adopts an **Entity-Component-Worker** model.

- **Entities** are anything in the world that have a position.
- **Components** define state and how other entities interact with them.
- **Workers** are micro-services that simulate components of entities in the world.

2.8.3 Developer Tools

There is extensive logging of performance metrics, which is a fantastic starting point for scaling up a simulation[8]. In addition to this, there is an active developer **Forum** - where SpatialOS developers and Improbable engineers meet to help other SDK users solve their problems[17].

Naturally as the public SDKs have matured, so too have the public-facing developer tools. Currently, there is an **Inspector** to view the location of entities and the values that their components have at a given time, in either a local or deployed simulation.

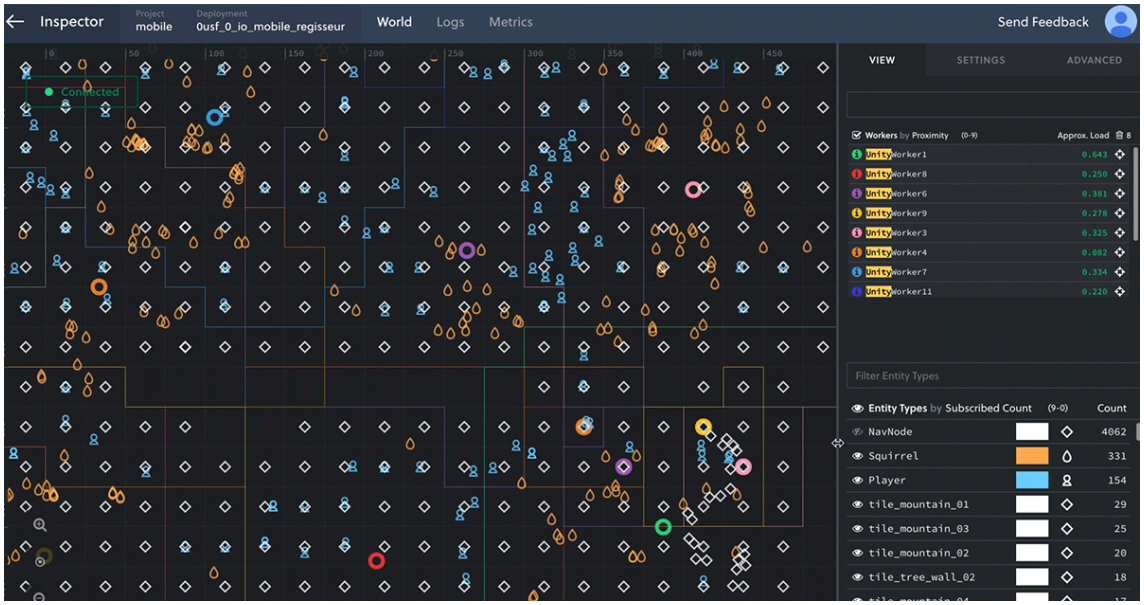


Figure 18: The Inspector demonstrating a deployment of the iOS Demo, *Quest*. [19]

2.8.4 Layered Simulation

By virtue of games having several dynamic systems all interlinking with each other, it is possible to create layered simulations on SpatialOS. This works well with the concept of Global, Reactive and Zonal layers that have been mentioned already in Sections 2.4, 2.5 and 2.6. Furthermore, as SpatialOS is distributed by nature, it is the ideal platform to create and execute massive layered, distributed simulations.

3 Prioritising Economic Value

Thus far, we have shown that whilst there is great scope to utilise drones for delivery of physical goods, the existing implementation of AATC only touches upon the routing aspect - how to get from point A to point B. Even in all the test cases, the drones' start and end points were hand-picked by the developers[2] to give an insight into how AATC would operate in a variety of scenarios.

To gain greater understanding of how drones can positively impact industries, simulations need to be performed which take into account the financial aspect of their use case. For example when it comes to drone delivery networks, we want to ensure that the drones not only reach their destination quickly, but that they also get scheduled and utilised in the appropriate manner to maximise the profitability of the network.

3.1 Quality of Service Value Curve

In order to consider how to maximise profitability, we introduce the concept of a Value Curve related to the Quality of Service (QoS) provided. Rather than deal with the fixed pricing model of delivery fees as we see in the current day, we propose a new mechanism where the cost of the delivery is modelled on a curve closely related to the QoS.

4 Project Plan

4.1 Overview

We envision the project to have three major development phases:

1. Porting the Global and Reactive Layers to SpatialOS.
2. Implementing the delivery Scheduling Layer.
3. Implementing tools to visualise the financial value in the simulation.

Once this has been complete we wish to conduct simulations with different scheduling models, such as First Come First Serve (FCFS) and the QoS-driven scheduler, to compare their total profits or loss. After this analysis is complete, we will compile the relevant data and work on the final project report.

4.2 Stretch Goals

If the project is ahead of schedule, we plan to:

- Introduce static toll zones.
- Introduce dynamic no fly zones.
- Introduce dynamic toll zones.
- Investigate the drone delivery network use-case of blockchain.
- Implement the Scheduling Layer as a custom worker.
- Implement the Reactive Layer as a custom worker.
- Implement the Global Layer as a custom worker.

4.3 Timeline

Week Beginning	Project Phase	Task
29th January	1	Global Layer
5th February	1	Global Layer
12th February	1	Global Layer
19th February	1	Global Layer
26th February	1	Global + Reactive Layer
5th March	1	Reactive Layer
12th March	1	Reactive Layer
19th March	1	Break for Exams
26th March	1	Reactive Layer
2nd April	2	Scheduling Layer Tooling
9th April	2	Scheduling Layer Tooling
16th April	2	Scheduling Layer
23rd April	2	Scheduling Layer
30th April	2	Scheduling Layer
7th May	3	Visualiser Tooling
14th May	3	Visualiser Tooling
21th May	3	Visualiser Tooling
28th May	3	Visualiser Tooling
4th June	Analysis	Run Simulations and Collect Data
11th June	Report	Draft a Final Report
18th June	Report	Finalise Report

Table 4: Project Timeline

5 Evaluation Plan

The minimum critera for the project is for a SpatialOS simulation to demonstrate drones being scheduled, by any possible mechanimm, and subsequently routed in a similar manner to the AATC project.

It would be beneficial to be able to show that simulations have been run using different drone scheduling algorithms, especially the approach we have introduced that is based on the QoS value curve. Here they are, listed:

- First Come First Serve
- Shortest Job First
- Priority
- Multi-Level Feedback Queue
- Quality of Service Value Curve

When running the above simulations, key financial figures such as revenue, loss and profit should be logged and recorded to be analysed. The project can be considered a success if there is clear breakdown and analysis of these key variables, and how they compare across different scheduling mechanisms.

After letting a few test users play around on a few simulations, we will ask a few short questions on how they found the experience. By allowing users to control locations of No Fly Zones, the number of drones in the sky, and easily switch between scheduling algorithms on-the-fly, we would hope for users to find the entire simulation more visceral and engaging.

Instead of trusting that we are being honest about our findings, showing and letting users discover it for themselves should prompt them to understand the entirety of this project better. This can be validated by looking at the survey results.

We also plan on using the survey as a mechanism to get feedback on user-focused aspects the project, such as ease of use. This would help to conduct a better evaluation by having actual data that can be used to determine what was received well and what was received poorly. In turn, we could then identify what future work needs to be done to take the project further in the future.

References

[1] Paul Balaji, David Cattle, Andrea Janoscikova, Galina Peycheva, Jan Matas, and Samuel Wood. *AATC Visualiser*. 2017. URL: https://samuknet.github.io/aatc-visualiser/ (visited on 01/24/2018).

[2] Paul Balaji, David Cattle, Andrea Janoscikova, Galina Peycheva, Jan Matas, and Samuel Wood. *Autonomous Air Traffic Control*. Tech. rep. 2017, p. 39.

[3] BBC News. *'Drone' hits BA plane: Police investigate Heathrow incident*. 2016. URL: http://www.bbc.co.uk/news/uk-36069002.

[4] BBC News. *First UK police drone unit launched in Devon, Cornwall and Dorset*. 2017. URL: http://www.bbc.co.uk/news/uk-england-devon-40595540 (visited on 01/25/2018).

[5] BBC News. *Passenger jet approaching Heathrow in drone 'near-miss'*. 2017. URL: http://www.bbc.co.uk/news/uk-england-london-39457371 (visited on 01/23/2018).

[6] John Bell. *CPU Scheduling*. 2018. URL: https://www.cs.uic.edu/%7B~%7Djbell/CourseNotes/OperatingSystems/5%7B%5C_%7DCPU%7B%5C_%7DScheduling.html.

[7] Owen Bowcott. *Laws for safe use of driverless cars to be ready by 2021*. 2017. URL: https://www.theguardian.com/law/2017/dec/14/laws-safe-use-driverless-cars-ready-2021-law-commission.

[8] Callum Brighting. *Crowd Sim In SpatialOS: How to scale a deployment?* 2017. URL: https://forums.improbable.io/t/crowd-sim-in-spatialos-how-to-scale-a-deployment/2547 (visited on 01/17/2018).

[9] Rob Davies. *Drone flew 'within wingspan' of plane approaching Heathrow*. 2017. URL: https://www.theguardian.com/technology/2017/mar/31/drone-wingspan-plane-approaching-heathrow-near-misses.

[10] Clay Dillow. *This unsexy technology is set to revolutionize the drone industry*. 2015. URL: http://fortune.com/2015/05/05/drone-technology/.

[11] Flightradar24. *About Flightradar24*. 2018. URL: https://www.flightradar24.com.

[12] Mirmojtaba Gharibi, Raouf Boutaba, and Steven L. Waslander. "Internet of Drones". In: *IEEE Access* 4 (2016), pp. 1148–1162. ISSN: 21693536. DOI: 10.1109/ACCESS.2016.2537208. arXiv: 1601.01289.

[13] Samuel Gibbs. *Uber plans to buy 24,000 autonomous Volvo SUVs in race for driverless future*. 2017. URL: https://www.theguardian.com/technology/2017/nov/20/uber-volvo-suv-self-driving-future-business-ride-hailing-lyft-waymo.

[14] Google. *"Drone Technology" Google Trends*. 2018. URL: https://trends.google.co.uk/trends/explore?date=all%7B%5C&%7Dq=drone%20technology.

[15] Kristen Hall-Geisler. *Google Gets a Patent for an Autonomous Delivery Truck*. 2016. URL: https://www.popsci.com/google-gets-patent-for-an-autonomous-delivery-truck (visited on 01/25/2018).

[16] Kristin Hohenadel. *Skype Co-Founders Want to Overhaul Local Delivery With Sidewalk Robots*. 2015. URL: http://www.slate.com/blogs/the%7B%5C_%7Deye/2015/11/02/starship%7B%5C_%7Dtechnologies%7B%5C_%7Dis%7B%5C_%7Dbuilding%7B%5C_%7Dself%7B%5C_%7Ddriving%7B%5C_%7Dsidewalk%7B%5C_%7Drobots%7B%5C_%7Dthat%7B%5C_%7Dwill.html.

[17] Improbable Worlds Ltd. *Improbable Forums*. 2018. URL: https://forums.improbable.io/.

[18] Improbable Worlds Ltd. *Improbable Home Page*. 2018. URL: https://improbable.io/.

[19] Improbable Worlds Ltd. *Quest- an iOS io game from Improbable*. 2017. URL: https://improbable.io/games/blog/quest-an-ios-io-game-from-improbable-io (visited on 11/12/2017).

[20] Improbable Worlds Ltd. *SpatialOS Technical Breakdown*. 2018.

[21] Improbable Worlds Ltd. *What is SpatialOS?* 2018. URL: https://docs.improbable.io/reference/12.0/shared/concepts/spatialos%7B%5C#%7Ddeveloping-with-spatialos.

[22] Improbable Worlds Ltd. *What we found when we simulated the backbone of the entire Internet on SpatialOS*. 2016. URL: https://improbable.io/company/news/2016/03/24/what-we-found-when-we-simulated-the-backbone-of-the-entire-internet-on-spatialos.

[23] Mark Jarman, John Vesey, and Paul Febvre. "UAVs for UK Agriculture : Creating an Invisible Precision Farming Technology - White Paper". 2016. URL: https://sa.catapult.org.uk/wp-content/uploads/2016/07/White-paper-UAVs-and-agriculture%7B%5C_%7DFinal2.pdf.

[24] Martin Joerss, Florian Neuhaus, Christoph Klink, and Florian Mann. "Parcel delivery The future of last mile". In: September (2016). URL: https://www.mckinsey.com/%7B~%7D/media/mckinsey/industries/travel%20transport%20and%20logistics/our%20insights/how%20customer%20demands%20are%20reshaping%20last%20mile%20delivery/parcel%7B%5C_%7Ddelivery%7B%5C_%7Dthe%7B%5C_%7Dfuture%7B%5C_%7Dof%7B%5C_%7Dlast%7B%5C_%7Dmile.ashx.

[25] Kia Kokalitcheva. *This Cute Self-Driving Robot Wants to Deliver Your Food or Laundry*. 2016. URL: http://fortune.com/2016/04/06/dispatch-carry-delivery-robot/ (visited on 01/25/2018).

[26] Robin Koontz. *Drones to the Rescue*. 2014. URL: http://www.kidsdiscover.com/teacherresources/drones-uavs-rescue/ (visited on 01/24/2018).

[27] Mike Laris. *Driverless delivery robots could be hitting D.C. sidewalks soon.* 2016. URL: http://www.chicagotribune.com/bluesky/technology/ct-driverless-delivery-robots-20160328-story.html (visited on 01/25/2018).

[28] Hau L. Lee, Yiwen Chen, Barchi Gillai, and Sonali Rammohan. "Technological Disruption and Innovation in Last-Mile Delivery". In: *Stanford Business* June (2016), pp. 1–26. URL: https://www.gsb.stanford.edu/sites/gsb/files/publication-pdf/vcii-publication-technological-disruption-innovation-last-mile-delivery.pdf.

[29] Prashob Menon. *HOW MANY DRONES DOES AMAZON NEED?* 2013. URL: http://iveybusinessreview.ca/blogs/pmenonhba2010/2013/12/11/how-many-drones-does-amazon-need/.

[30] Melanie Mitchell. "An introduction to genetic algorithms". In: *Computers & Mathematics with Applications* 32.6 (1996), p. 133. ISSN: 08981221. DOI: 10.1016/S0898-1221(96)90227-8. URL: https://books.google.ca/books/about/An%7B%5C_%7DIntroduction%7B%5C_%7Dto%7B%5C_%7DGenetic%7B%5C_%7DAlgorithms.html?id=0eznlz0TF-IC%7B%5C&%7Dpgis=1%7B%5C%%7D5Cnhttp://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20%7B%5C&%7Dpath=ASIN/0262631857%7B%5C%%7D5Cnhttp://linkinghub.elsevier.com/retrieve/pii/S0898122196902278.

[31] Jussi Myllymaki. *Autonomous Delivery Platform*. 2013. URL: http://pdfpiw.uspto.gov/.piw?PageNum=0%7B%5C&%7Ddocid=09256852%7B%5C&%7DIDKey=EA66D62BF248%7B%5C%%7D0D%7B%5C%%7D0A%7B%5C&%7DHomeUrl=http%7B%5C%%7D3A%7B%5C%%7D2F%7B%5C%%7D2Fpatft.uspto.gov%7B%5C%%7D2Fnetacgi%7B%5C%%7D2Fnph-Parser%7B%5C%%7D3FSect1%7B%5C%%7D3DPTO2%7B%5C%%7D2526Sect2%7B%5C%%7D3DHITOFF%7B%5C%%7D2526p%7B%5C%%7D3D1%7B%5C%%7D2526u%7B%5C%%7D3D%7B%5C%%7D25252Fnetahtml%7B%5C%%7D25252FPTO%7B%5C%%7D25252Fsearch-bool.html%7B%5C%%7D2526r%7B%5C%%7D3D25%7B%5C%%7D2526f%7B%5C%%7D3DG%7B%5C%%7D2526l%7B%5C%%7D3D50%7B%5C%%7D2526co1%7B%5C%%7D3DAND%7B%5C%%7D2526d%7B%5C%%7D3DPTXT%7B%5C%%7D2526s1%7B%5C%%7D3DGoogle.ASNM.%7B%5C%%7D2526OS%7B%5C%%7D3DAN%7B%5C%%7D2FGoogle%7B%5C%%7D2526RS%7B%5C%%7D3DAN%7B%5C%%7D2FGoogle.

[32] Herman Narula. *Google Cloud Next '17 — Herman Narula Day 3 Keynote*. 2017. URL: https://youtu.be/fTuijWHwAsk?t=4m46s.

[33] A. Nash, S. Koenig, and C. Tovey. "Lazy Theta *: Any-Angle Path Planning and Path Length Analysis in 3D". In: *Third Annual Symposium on Combinatorial Search (SOCS-10) Lazy* (2010), pp. 153–154.

[34] Jonathan Roberts and Unlike Formula. *What is FPV drone racing ?* 2016. URL: https://phys.org/news/2016-02-fpv-drone.html.

[35] Lynn Root. *Python implementation of Dijkstra's Algorithm*. URL: https://gist.github.com/econchick/4666413 (visited on 01/25/2018).

[36] Aaron Souppouris. *Skype co-founders build delivery bot that rides on sidewalks*. 2015. URL: https://www.engadget.com/2015/11/02/starship-technologies-local-delivery-robot/ (visited on 01/25/2018).

[37] Peter Stearns. *The Encyclopedia of world history : ancient, medieval, and modern, chronologically arranged*. Boston: Houghton Mifflin, 2001. ISBN: 0395652375.

[38] Tesla Inc. *Tesla Autopilot*. 2018. URL: https://www.tesla.com/en%7B%5C_%7DGB/autopilot (visited on 01/25/2018).

[39] Tansel Uras and Sven Koenig. "An Empirical Comparison of Any-Angle Path-Planning Algorithms". In: *Aaai* (2015). DOI: 10.1002/bse.

[40] Richard Verrier. *Drones are providing film and TV viewers a new perspective on the action*. 2015. URL: http://www.latimes.com/entertainment/envelope/cotown/la-et-ct-drones-hollywood-20151008-story.html.

[41] Dan Wang. *Drone Delivery Economics*. URL: https://www.flexport.com/blog/drone-delivery-economics.

[42] Adrienne Welch. "A cost-benefit analysis of Amazon Prime Air A Cost-Benefit Analysis of Amazon Prime Air". In: (2015), p. 57.

[43] Lihua Zhu, Xianghong Cheng, and Fuh-Gwo Yuan. "A 3D collision avoidance strategy for UAV with physical constraints". In: *Measurement* 77.Complete (2016), pp. 40–49. DOI: 10.1016/j.measurement.2015.09.006.

Appendices

A Python Implementation of Dijkstra's Algorithm

```
def dijsktra(graph, initial):
  visited = {initial: 0}
  path = {}

  nodes = set(graph.nodes)

  while nodes:
    min_node = None
    for node in nodes:
      if node in visited:
        if min_node is None:
          min_node = node
        elif visited[node] < visited[min_node]:
          min_node = node

    if min_node is None:
      break

    nodes.remove(min_node)
    current_weight = visited[min_node]

    for edge in graph.edges[min_node]:
      weight = current_weight + graph.distance[(min_node, edge)]
      if edge not in visited or weight < visited[edge]:
        visited[edge] = weight
        path[edge] = min_node

  return visited, path
```

Listing 1: Example Implementation from GitHub. [35]