Joseph Law McGuchan

Modeling and Visualization of Networking in Low Earth Orbit Constellations

Computer Science Tripos – Part II

King's College

May 17, 2019

Declaration

I, Joseph Law McGuchan of King's College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

I, Joseph Law McGuchan of King's College, am content for my dissertation to be made available to the students and staff of the University.

Signed Joseph McGuchan

Date 17/05/2019

Proforma

Candidate Number:	**2345A**
Project Title:	**Modeling and Visualization of Networking in Low Earth Orbit Constellations**
Examination:	**Computer Science Tripos – Part II, May 2019**
Word Count:	**8231**
Line Count:	**2210**
Project Originator:	Dr Timothy Griffin
Supervisor:	Dr Timothy Griffin

Original Aims of the Project

To develop multiple routing algorithms for the Starlink network, a constellation of satellites to be created by the SpaceX company with the intention of providing low-latency internet connections across the Earth. To compare these algorithms on latency, fault tolerance, and response to high demand. And to create visualizations of these routing algorithms at work.

Work Completed

Routing algorithms have been developed and tested on all three properties. Response to high demand was not sufficiently tested tho conclusions about it have been made. Visualizations of the routing algorithms at work have been created.

Special Difficulties

SpaceX announced numerous significant changes to their network during the production of this dissertation which has lead to a number of differences.

Contents

List of Figures

Chapter 1

Introduction

SpaceX are planning to launch a constellation of thousands of low Earth orbit (LEO) communication satellites in the next few years. The objective of this constellation, called Starlink, is to provide low-latency internet connection across the world.

The satellites in this network will be in constant motion, not just relative to the ground, but relative to one another, creating a network with a constantly changing topology and associated latencies. The question of how to structure such a network, and what the resultant latencies of said network will be, has not been thoroughly explored, but it will become increasingly relevant as more and more companies build similar constellations. If these Ccnstellations prove to provide significant gains in latency while providing competitive bandwidth, they might render previous submarine optical cables obsolete.

My goals are:

1. To create visualizations of the SpaceX constellation.
2. To experiment with different topologies of the SpaceX network and test their associated latencies.

1.1 What is Starlink?

The purpose of Starlink is to provide low latency internet connections, as of 17/05/19, there are two companies offering satellite internet services, Excede[**?**], and Hughes, whose 9202 BGAN Land Portable Satellite Terminal offers connection speeds up to 464kbps[5]. These companies largely target domestic use in rural areas which don't have a faster coverage, and corporations, providing internet connections to airplanes and cargo ships. Currently, Satellite Internet connection is a last resort, something turned to when conventional means of connection are not available, Starlink intends to invert this, turning satellite internet into the premium option.

The difference between Starlink and currently existing brands is that current brands use geostationary satellites [11], while Starlink will use LEO satellites. The significantly shorter distance will create much shorter paths for signals. On top of this, SpaceX will be utilizing laser communication between satellites, as opposed to the competitors who have little to no intra-satellite communication. In the vacuum of space, light travels 47% faster than in glass [**?**]. Therefore, in theory, a LEO network utilizing lasers would achieve latencies far lower than that by even the best terrestrial fiber optic connections over long distances.

1.2 Why Focus on Starlink?

Starlink is only one of a number of different LEO internet networks that have been proposed, so why should I focus on it's topology? Ideally, I would like the conclusions of my study to be generalized to many other LEO constellations. However, constellations are approved by the FCC on a case-by-case basis, using a complicated and changing system of legislation, it's not easy to know what a normal network looks like. By analyzing the properties of a network of my own design, I run a much greater risk of coming to conclusions that cannot be generalized, as I am not working on an FCC-approved constellation. By analyzing the properties of Starlink, we are analyzing a network that is confirmed legally and scientifically plausible.

But there is another reason to investigate Starlink. As the largest of most high-profile attempts to build a LEO internet backbone, Starlink represents the most significant competitor to other emerging satellite communications networks and the one more likely to become dominant in future years. By theorizing about it's properties now, we can prepare ourselves for the changes Starlink might pose to communications in the upcoming years.

1.3 Why Create a Visualisation?

When it comes to understanding a network such as Starlink, a visual description is incredibly valuable. By visualizing the network we can develop an intuition for how it operates, and use that intuition to develop ideas for new algorithms and structures for testing.

Creating a visualization also poses a minimal additional cost on my part, as to accurately model the latencies between base stations I will need to simulate satellite positions and links anyway.

1.4 Mark Handley's Work

My initial proposal for the project was inspired by the findings of an existing study done by Mark Handley, since proposing the project, Mark Handley has made his results publicly available [10], so my new goal will be to replicate and expand upon his findings.

1.5 What I Will Be Using

To create the visualization I will be using the open-source game engine Godot. Using a game engine struck me as the simplest way to create a visualization tool, and Godot, being powerful, open-source, and capable of running easily on many devices, seemed like the ideal choice.

For the parts of code related to visualization I will be using Godot's build in script GDScript, which is easy to use and specifically designed to integrate well with the Godot engine. For parts of the code related to simulation I will use C#, which Godot is compatible with, and which offers significance performance improvements over GDScript.

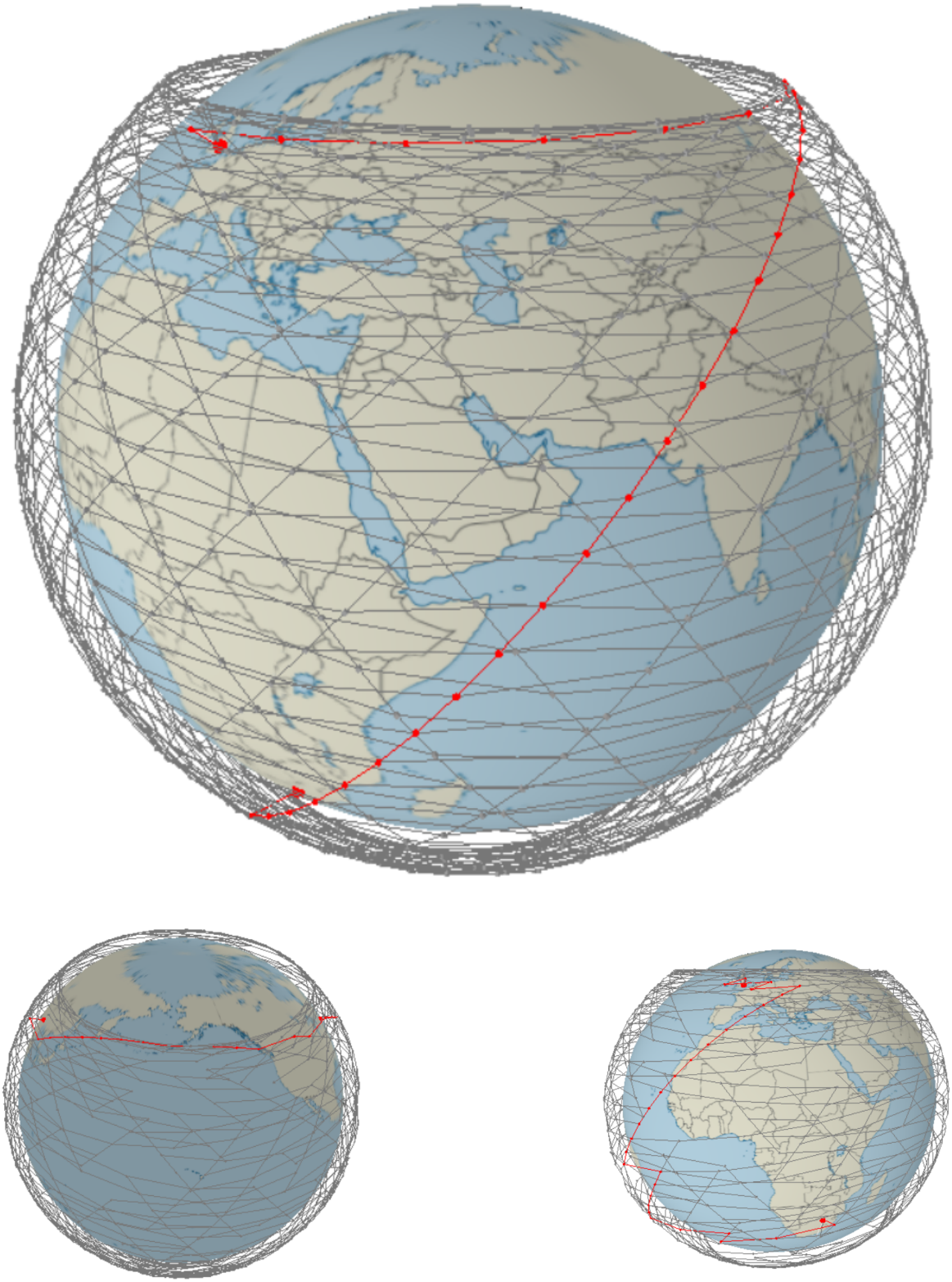
1.6 Example of Results

To give an example of my program at work I have included a few screenshots seen in figure 1.1. The program that I created is capable to simulating different linking methods between satellites, different rates of satellite failure, and performing a number of tests including highlighting the shortest path between two nodes. In all the images the red line is the shortest path between the two spherical base stations, while the gray lines represent available links.

1.7 Current State of Events

SpaceX have already sent up two test satellites, and according the Elon Musk, they are working very well, providing a latency of only 25ms[6], though Elon Musk neglected to specify which two locations the 25ms latency was between. As of May 11th 2019, SpaceX have loaded their first 60 satellites into their Falcon rocket, and are preparing to launch them [**?**]. This report is therefore being released at the optimal time for researchers hoping to get ahead of what could potentially be the next revolution in internet communications.

Figure 1.1: Screenshots of my program simulating the shortest path between two nodes through different linking methods and rates of satellite failure, the highlighted path is the shortest path between the two base stations.



Chapter 2

Preparation

2.1 Starting Point

Most of the information I will be using for the structure of the SpaceX network comes from their application to the FCC on March 29th, 2018[8], and their technical attachment[9], which outlines the various orbital spheres, planes within each sphere, and various other details of the constellation. I will also be building on insights gleaned from Handley's original research. The insights that I will be building off of are:

1. The optimum phase offset is 9/33.
2. Satellites will most likely link to those moving in the same direction as them.
3. Due to highly predicable nature of the network, Starlink will most likely use a static routing method, with a path for a packet being precomputed at the ground stations. It is therefore more useful to look at the shortest paths through the network isntead of simulating a routing method in action.

These insights will be returned to throughout my report and explanations will be given as to exactly what these insights mean.

2.2 The Structure of Starlink

There is a lot we do not know about Starlink, but this is what we can infer from SpaceX's application to the FCC[8],

Starlink was initially going to be a constellation of 4,425 LEO satellites. The initial proposal to the FCC came in the form outlined in figure 2.1, however Starlink have since changed their proposed satellite constellation[17]. It is as of yet unclear what structure the completed Starlink network will have, but we do know what the first sphere's properties will be (given in figure 2.4).

In it's application to the FCC SpaceX was required to state any notable debris that might not burn up if a satellite were to descend from orbit. Among the components listed were 4 silicon carbide "communication components", silicon carbide is used in mirrors for laser communication links, and we can therefore conclude that SpaceX's satellites will have, at most, 4 available links to form with nearby satellites.

We also know information about how Starlink satellites will communicate with ground stations, a base station may only connect with a satellite at an inclination less than

Figure 2.1: The layout of Starlink initially proposed

SPACEX SYSTEM CONSTELLATION					
Parameter	Initial Deployment (1,600 satellites)	Final Deployment (2,825 satellites)			
Orbital Planes	32	32	8	5	6
Satellite per Plane	50	50	50	75	75
Altitude	1150km	1110km	1130km	1275km	1325km
Inclination	53°	53.8°	74°	80°	70°

40°from the vertical using Radio Frequency (not laser) connections. Connections will be stronger to satellites directly overhead, however as Starlink does not provide much information in exactly how their intra-satellite communications work, speculations on bandwidth will not prove to be helpful.

2.3 Modeling Satellites in Orbit

The position of a satellite above the surface of the earth can be described by 4 values. [15]

Altitude

Inclination This is the angle between the orbital plane, and the equatorial plane, (the plane on which the equator lies).

Longitudinal Offset If inclination is greater than 0°, then the orbital plane and equatorial plane will intersect at a line, the angle between this line and the plane described by the great circle at longitude 0°is the longitudinal offset. In other words, the Longitudinal offset is the Longitude of the point where this line passes through the surface of the earth.

True Anomaly The angle between the line drawn from the origin to the satellite, and the line drawn from the origin to the point on the satellite's orbit of longitude 0.

To better visualize this, figure 2.2 describes how would would calculate the position of a satellite in space from these four values.

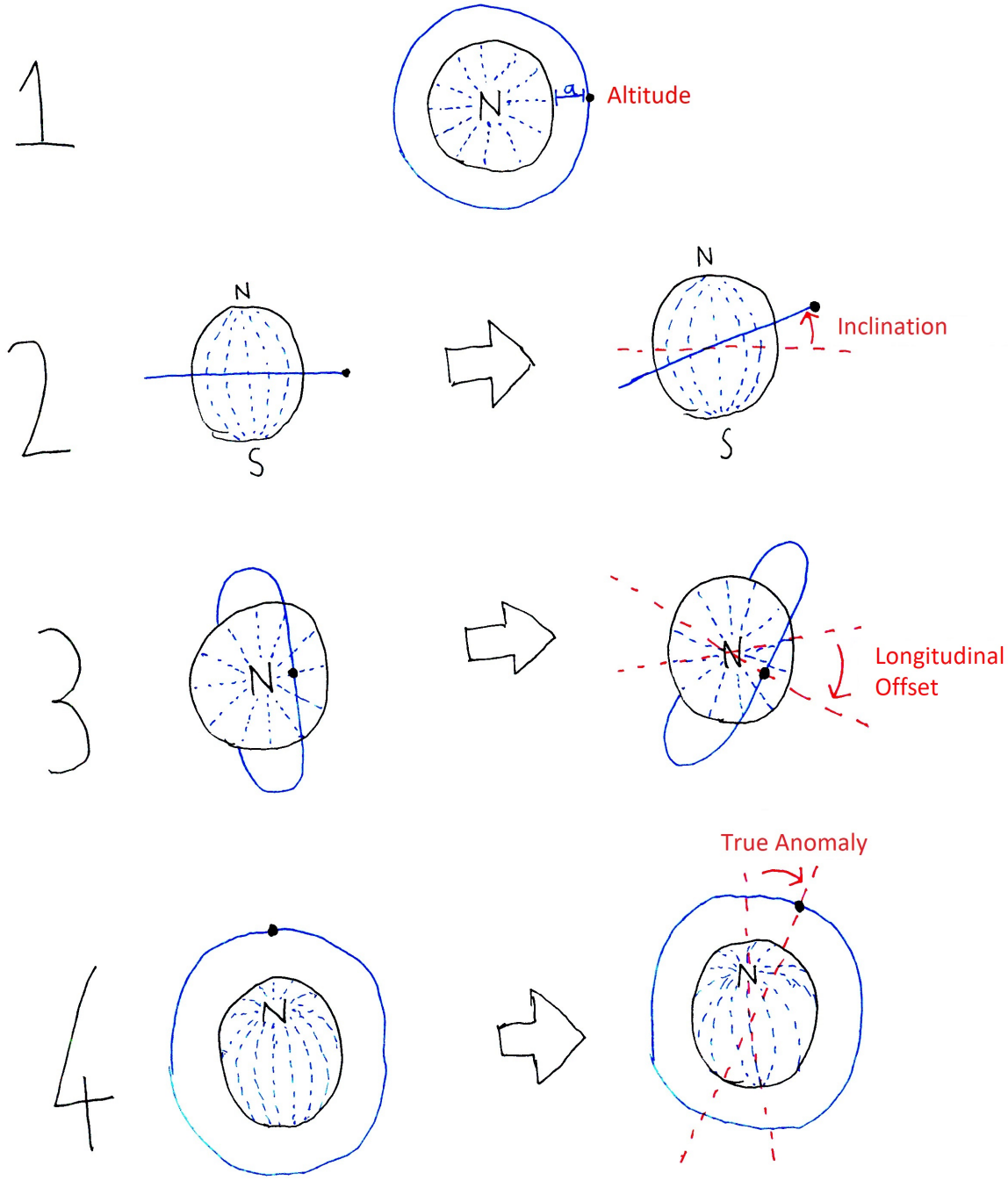
1. First, take the orbit around the equator of the chosen altitude. Place your satellite so that it is directly above the position with terrestrial coordinates (0,0).

2. Now, look at the earth from the side, so that your satellite is as far to the right of your view as possible, and the orbit it is on is flat. Tilt to the orbit until it's angle to the equator is equal to the inclination.

3. Now return to looking down on the earth from above, and rotate the orbit by your longitudinal offset clockwise.

4. Finally move so that the orbit forms a flat circle in your view, rotate the satellite clockwise around it's orbit by the true anomaly.

Figure 2.2: How to calculate the location of a satellite from it's Altitude, Inclination, Longitudinal Offset and True Anomaly



Note that the altitude, inclination, and longitudinal offset together describe a circular orbit, with the true anomaly then specifying a location on this orbit to place our satellite.

Also not that this only describes circular orbits. There is another factor called eccentricity that describes how "squashed" the orbit is in shape. For the purpouse of this model we will assume eccentricity to be 0, as SpaceX have not applied for a high-eccentricity orbit[8], and 0 eccentricity orbits are significantly easier to model (see below).

With these 4 values can describe the position of a satellite in space and its orbit, they do not describe how the satellite moves. For this we need two more variables.

Initial Anomaly This is the satellite's true anomaly at time 0.

Retrograde Status A Boolean variable. A retrograde orbit is one that goes against the rotation of the earth, typically these are described by orbits with a longitudinal offset greater than 180°. It is significantly more expensive to put a satellite into a retrograde orbit, and is essentially never done.[15]

Together, the altitude, inclination, longitudinal offset, initial anomaly and retrograde status uniquely determine a function that takes a time, and returns the position in 3D space that our satellite will be occupying at that time. It does this by first calculating the true anomaly, and then applying the transformations detailed in figure 2.2 to translate this into a location. The rate of change of true anomaly, the angular velocity, is given by:

$$\sqrt{\frac{GM}{r^3}}$$

Where r is the distance from the origin, or altitude + radius of the Earth.

Note that while altitude is constant (eccentricity = 0) velocity is unchanging, and true anomaly can be described as a linear function of time.

$$TrueAnomaly(t) = InitialAnomaly + t\sqrt{\frac{GM}{r^3}}$$

Or $TrueAnomaly(t) = InitialAnomaly - t\sqrt{\frac{GM}{r^3}}$ if the orbit is retrograde.

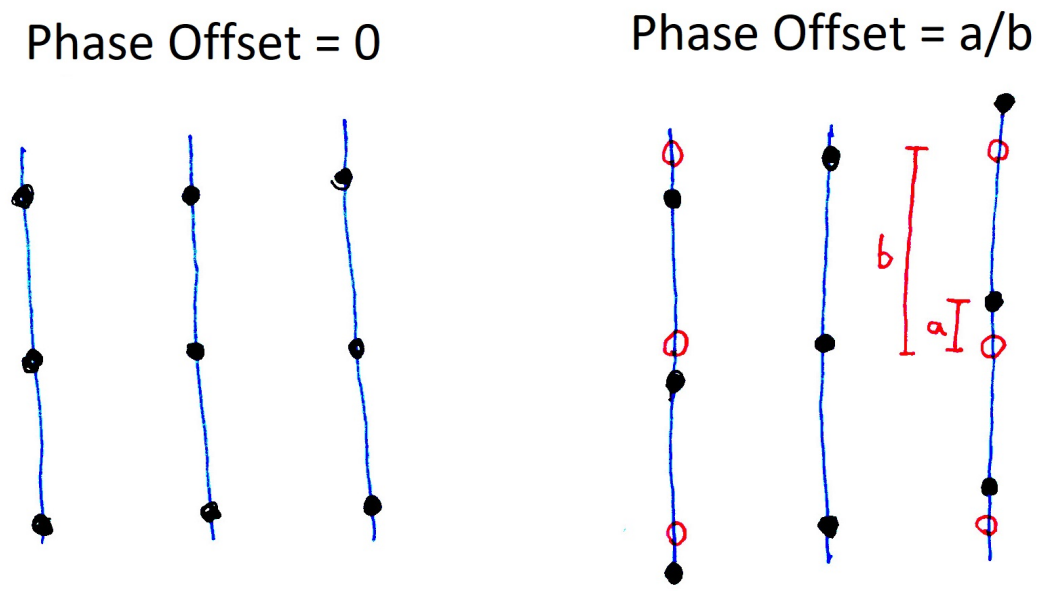
Because the formula for calculating the position of a satellite at a given time is so easy to compute, we can calculate these values in advance, and then linearly interpolate between a series of fixed points to approximate the satellite motion. This means we do not run the same risks normal discrete physics models face when describing continuous behavior, such as the energy of the system changing, or orbital precession (the point of furthest distance moving steadily), as we will not be approximating a continuous process from a series of discrete steps but instead will be taking samples from a continuous function.

For the rest of this document, a satellite will be denoted by $x_{i,j,k}$. Describing the kth satellite on the jth orbital place in the ith orbital sphere. L gives the number of orbital spheres, with M_i giving the number of orbits in orbital sphere i, and N_i, j giving the number of satellites in the jith orbit of the ith sphere. The jth orbit in the ith sphere will be denoted $o_{i,j}$, and the ith sphere will be denoted s_i

2.4 Unknown Variables

While we can learn a lot from SpaceX's applications, there are still a number of variables that are left undetermined.

Figure 2.3: Phase Offset described by looking at three adjacent orbits.



Phase Offset As we do not know for certain how satellites will be positioned in orbits, we also do not know how they will be positioned relative to other orbits.

Sphere Offset As well as satellites being offset from one another, whole spheres can be offset too.

Link usage Each satellite will have a maximum of 4 links, however, we do not as of yet know which arrangements of links are the most optimal.

I will only be looking at a single sphere in my study, so sphere offset will not need to be considered. In my implementation I will go through how I described linking methods.

2.5 Phase Offset

In Mark Handley's paper he went through a method of describing phase offset that I will use in my implementation, outlined in figure 2.3. Here we are looking at three adjacent orbits with a phase offset of 0, and then with a phase offset of a/b, the empty red circles give where a satellite would have been if the phase offset was 0. As you can see the phase offset is a value between 0 and 1 that describes the displacement of satellites in each orbit relative to one another. With 0 meaning that satellite $x_{0,0,0}$ crosses the equator at the same time as $x_{0,1,0}$, 1 meaning that $x_{0,0,0}$ crosses at the same time as $x_{0,1,1}$, and other values being a linear interpolation between these two states.

Because the constellation wraps around the earth, with $o_{i,M_i} = o_{i,0}$, it is necessary that the phase offset be a multiple of $1/N_{i,0}$, so that the accumulated phase offset at o_{i,M_i} results in an arrangements of satellites identical to $o_{i,0}$. In Mark Handley's research, he the optimum for the Starlink network to be 9/33. Which I will use in my research.

All the details that we know about the Starlink constellation going into implementation are given in figure 2.4. What will need to be determined upon implementation is how to describe the linking method.

Figure 2.4: An example for how a constellation might be described.

Number of Orbital Spheres	1
Altuitude	550km
Inlication	53°
Number of Orbits	24
Sattellites per Orbit	66
Phase Offset	9/66

Chapter 3

Implementation

3.1 Coding for Visualisation and Efficent Algorithms

This implementation chapter is divided into two main sections, the first section will go over how I represented the constellations and the code I wrote to perform tests on this model, the second section will go over the executable file, it's structure, and how to use it. While these sections are separate in my report, they were developed concurrently, and are connected to each other. As I visualized the constellation, I was able to gain a better understanding of it's structure, this directed the properties of the network I wanted to examine, which in turn effected the tests I wanted to create and so the algorithms I needed to implement. Furthermore as Godot is designed to model 3D environments it provides a number of functions that in turn influenced how I wrote my algorithms, most notably Godot provides a simple framework for multi-threading through a _process function that any class extending Node can override and which is called on every step by the engine. Godot handles multi-threading of the _process function internally. This gave me the opportunity to exploit threading to improve the efficiency of the program without having to do any extra work. All that said, I think to is clearer to divide these two halves of the program when talking about my implementation.

After these two sections will be one more outlining observations that were made of the network's behavior during my experimenting on it, and how these observations lead to further development in my linking algorithms.

3.2 Modeling the Network

3.2.1 Patterns used

Due to the extensive combustibility of the network, I elected to use the Template Method pattern in three separate locations.

- To describe different linking methods.
- To describe various different ways of colorizing the network to highlight certain properties.
- To describe various tests to be applied to the network.

I also utilized a variant of the Builder pattern for describing constellations and tests. The ConstellationDescription class is initialized with a description of a constellation similar to what was given above, including the tests and coloring method I want to use on the class. This allowed me to easily categorize and load new constellations.

3.2.2 Modeling Satellite Positions

I initially did not precompute orbits but instead had every satellite recalculate it's position on every step. On implementation, I found that this was flawed, being able to simulate only 500 satellites in motion before slowing down. Because of this, I changed to a precomputed method. In this method, orbits are precomputed as a number of transformations, and the position of satellites is calculated by interpolating adjacent transformations, this sacrifices some accuracy, but vastly increases the computation speed. It is important that orbits be described by transformations (a combination of a point and rotation) so that satellites can be oriented to always have the earth below them from their reference frame. This makes a number of computations later on easier.

There are five overall classes for representing the different objects that make up a constellation. Here a "step" in a unit of in-sinulation time.

Constellation This is an overall class that contains a list of BaseStations and a list of OrbitalSpheres. Each step the constellation will run any programs that cannot be threaded, this include parts of the Linking Method, calculating the current rotation of the Earth, and any test that is being presently ran on the network.

BaseStation Describes a surface level station. Each step it moves to match the rotation of Earth.

OrbitalSphere Each step this updates an internal value known as "RotationOfOrbit". As all orbits in the same orbital sphere are at the same altitude, all the satellites in these orbits have the same angular momentum, it is therefore possible to store the rotation of all of these orbits in the same place to ensure they remain in sync. OrbitalSphere contains a list of Orbits.

Orbit An orbit contains a list of Satellites. It is the Orbit objects that do most of the work in calculating the positions of Satellites. They store a list of satellites along with a precomputed list of points that describe their orbit, each step they look at the RotationOfOrbit of their parent class and update the positions of the satellites in their list accordingly. It also calculates some of the information used for the LinkingMethod each step.

Satellite The Satellite class is a simple descriptor that stores information related to the satellite. Satellite objects do not posses a _process function that is called each step as this would be too costly for thousands of satellites to do.

Between them these classes can model the motion of an entire constellation with ground stations. Note however that they do not visualize it. To have something be visualized in Godot you have to add a "scene" containing a "node" to the "scene tree". A node can have associated scripts that they call the functions of, which is what these 5 classes are, but this it is worth bearing in mind that the "Satellite" class is not visualized, but rather instanced by a a node object called Satellite that is then visualized. I will go into how this works in the second section.

3.2.3 Representing Links

There are 3 primary ways to represent a graph for algorithms:

Adjacency Matrix An NxN matrix of values representing all possible connections. Obviously inefficient for a sparse set of connections like this.

Adjacency List Associate each Satellite with a list of the vertices it is connected to. Space efficient, but creates difficulty in running Dijkstra's algorithm.

Edge List Create a list of edges that refer to vertices. This is useful for algorithms that require iterating through all edges like Dijkstra's algorithm, but to find the adjacent satellites to any given satellite requires searching through the whole list.

Initially, I used adjacency lists, this lead to a number of problems.

- Some algorithms require a list of all edges, which is very hard to obtain with adjacency lists.
- When drawing edges in the visualization, each would be drawn twice, for each vertex they were associated with.
- When updating links, values such as the distance between two vertices had to be computed twice for each vertex.

Instead I created a new class of objects representing links, and gave each vertex a reference to the links associated with it.

In Godot, all drawn objects require one "parent" (different to class inheritance) which is already a drawn object. Children inherit the position and rotation of their parents, and rotating a group of objects is significantly more efficient than rotating a single object alone. Because of this, backwards and forward links, which do not change distance or position relative to the satellite, should be made children of the satellite, and they will then never have to be updated. Because of this, all links have a primary and secondary satellite. Which the link always being the child of the primary satellite.

3.2.4 Representing Linking Methods

As well as having a way to describe a link between two satellites I needed a way to describe the linking method used to form these links. To do this I used the Template Method pattern outlined above, however after experimenting with numerous different linking methods, I ended up settling on just two.

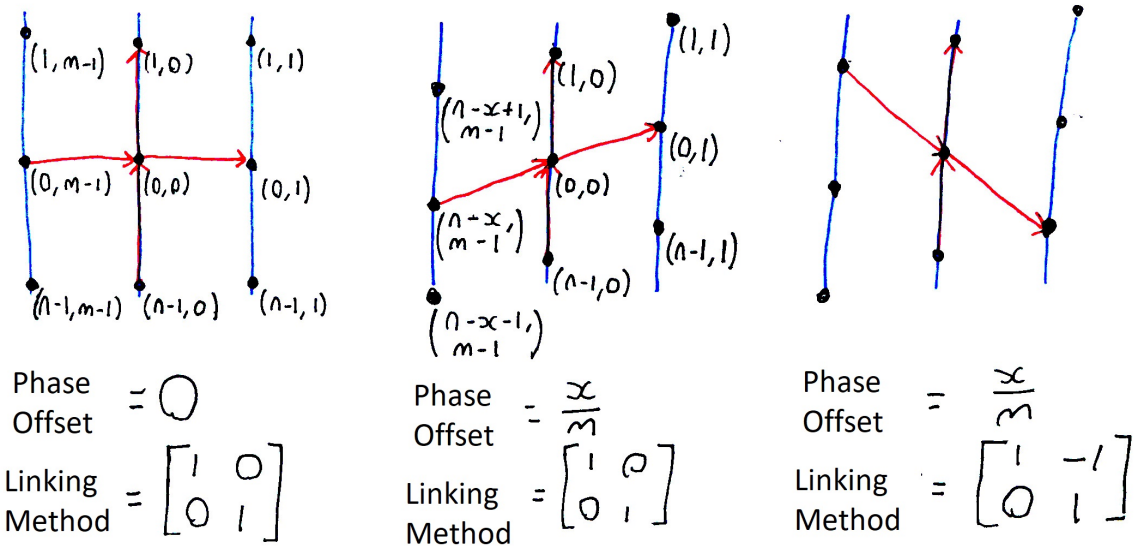
NoLinking

FixedLinking(X)

Fixed Linking is created with a matrix that can be customized to describe a huge range of potential linking methods. I will describe how this works below.

Fixed linking describes a linking method where links are constant, and each satellite links to it's neighbors in the same way, this creates a network with a grid-like topology. As links are bidirectional and all satellites have fixed, identical linking patterns, satellites

Figure 3.1: The relationship between the linking method used and phase offset.



must have an even number of links available to them, as the number of satellites they are "linking to" will be identical to the number that are "linking to them".

Fixed linking is initialized with a nx2 matrix, where n is half the number of links per satellite, and columns of the matrix correspond to the satellites that are linked to. Like so:

$$\begin{bmatrix} PositionsForwardinOrbit1 & PositionsForwardinOrbit2 \\ PositionstotheLeft1 & PositionstotheLeft2 \end{bmatrix}$$

So:

$$FixedLinking(\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Describes a linking method where each satellite has 6 links available, and connects to the satellite directly in front ($\begin{bmatrix} 1 \\ 0 \end{bmatrix}$), to the left ($\begin{bmatrix} 0 \\ 1 \end{bmatrix}$), and diagonally forward and to the left ($\begin{bmatrix} 1 \\ 1 \end{bmatrix}$), as well as forming connections with the three satellites it is in front of, to the left of, and forward-left of.

However, there is a difficultly here when it comes to phase offset. Because of the way phase offset accumulates between orbits, $x_{0,M_0-1,0}$ is not necessarily to the right of $x_{0,0,0}$. Because of this an adjustment must be made between o_{i,M_i} and $o_{i,o}$ to correct, as outlined in figure 3.1. Also I this figure I have given an example of a linking method with a negative value in one of it's positions.

I decided to dub linking methods of the form:

$$FixedLinking(\begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix}$$

As "Handley(x)" or "H(x)" linking methods, as they are the forms of linking that Mark Handley examined in his original paper.

3.2.5 Representing Tests

Tests were associated with a constellation and ran every step. To prevent slowing down, when a test ran it would check the system clock, and each second would run it's main code and produce outputs. Using the system clock might have posed a problem as in game time and system time might go our of sync, but Godot has inbuilt functionality to ensure that processes are kept in sync with system time, objects that are called with the _process function are passed a float that represents the amount of system time that has passed since the last _process call, by using this value in all my physics simulations I was able to guarantee that my program stayed in sync with system time.

All the tests had an associated file that they would output their results into, as well as logging them on the terminal, in csv format. Not all the computations were done by my code however, and some of the csv files had to be processed in spreadsheets to produce meaningful data.

3.2.6 Connected Components

One of my tests required me to calculate the number of connected components. This was done very simply through a number of depth-first searches starting at unvisited nodes until all nodes had been visited.

3.2.7 Dijkstra's Algorithm

I used Dijkstra's algorithm in order to calculate shortest paths around the network, with the modification to allow for quickly retrieving the shortest path when used. Somewhat surprisingly, C sharp doesn't come with a priority queue, so I used one sourced from 2 different tutorials. Taking the structure of one and then implementing the quick comparison operations from another.[1][2]

3.2.8 Calculating the Latency Across a Path

There are 4 factors that effect the latency of a path, some of which are easy to model, and others are significantly more tricky.

Processing delay The time it takes router to process the packet header, this is typically insignificant and will bee assumed to be 0.

Queuing delay The time the packet spends in routing queues, this is dependent on the load on the network and is therefore not easy to predict. I will be calculating latency assuming the best case scenario, which is no queuing delay.

Transmission delay The time it takes to push the packet onto the link. This is of course dependent on the size of the packet, but for the smallest packet sizes this value is negligible.

Propagation delay The time the signal spends in transmission, since all connections are of the form of electromagnetic waves this is simply the path length divided by the speed of light.

Figure 3.2: Keybindings.

Move Forward	W, Up Arrow
Move Backward	S, Down Arrow
Move Up	Q, Comma Key
Move Down	E, Full Stop Key
Move Left	A, Left Arrow
Move Right	D, Right Arrow
Restart Simualtion	R
Toggle Free Mouse	Esc
Next Scene	X
Previous Scene	Z

Because of this, the only calculation needed to calculate needed to get the latency of a path is to divide it's length by the speed of light in a vacuum. To calculate the round trip time (RTT), which is typically used when advertising linking systems, we simply multiply this value by two.

3.2.9 Efficiency vs Usability

My code went through about 2 major rewrites, and over the course of this I ended up using numerous abstract classes and wrappers that would simplify more complex objects. There are two kinds of vertex, base stations and satellites, and two kinds of link, those between satellites and those between a satellite and base station. Each of these are children of general Vertex and Link classes, and it is these classes that are used by the various tests and algorithms that run on the network. Initially my objective was to prioritize efficiency at all costs, but as I continued to develop the code, it became necessary to implement redundancies and inefficient storing methods, like lists, in order to make the code more easily usable. I also created a generic Graph object that contained only a list of Vertices and a list of Links, and a VertexParth object that allowed for easy manipulation and testing of paths.

3.3 Visualization

3.3.1 How to Use the EXE

If you would like to see the program that was used to generate these images and results, simply download and run the exe stored with the code.

The exe will create files for test results in the same location as it is ran, as well as printing to the command line the results of tests as they are written.

It is recommended that you run in Windows 10, as it is not guaranteed to work in other operating systems.

You can navigate the simulation using the mouse, along with arrow keys or WASD. Press R to reset the whole simulation. Use X and Z to move between different tests. Press ESC to toggle the mouse between captured (for moving the camera) and free.

All the tests run at 60x realtime, which I found to be a very good timefactor for observing overarching trends in the network.

There are a total of 17 scenes available. I will list them here, though at this point in the report it might not be clear what they mean.

1. Gradient Applied to Each Orbit
2. Following a Satellite and it's Neighbors
3. Handley(-1)
4. Handley(0)
5. DistantLinking
6. Connected Components with Deletetions: Handley(-1)
7. Connected Components with Deletetions: Handley(0)
8. Connected Components with Deletetions: DistantLinking
9. Shortest Path London-New York with Deletions: Handley(-1)
10. Shortest Path London-New York with Deletions: Handley(0)
11. Shortest Path London-New York with Deletions: DistantLinking
12. Shortest Path Beijing-New York with Deletions: Handley(-1)
13. Shortest Path Beijing-New York with Deletions: Handley(0)
14. Shortest Path Beijing-New York with Deletions: DistantLinking
15. Shortest Path London-Johannesburg with Deletions: Handley(-1)
16. Shortest Path London-Johannesburg with Deletions: Handley(0)
17. Shortest Path London-Johannesburg with Deletions: DistantLinking

Tests 6 to 17 will all run by themselves, moving over to the next scene once they are complete and outputting their results into a folder called TestResults. However it will take over 24 hours in total to run all of these tests and output their results.

3.3.2 How Godot Works

Godot works in a way similar to many other game engines. There is a "scene tree" that "nodes" can be on. Nodes may have associated "scripts". If a node on the scene tree has a script that inherits from the Node class and overrides the _process function, that _process function will be called every in game-step with a value representing the system time passes since the last call. Nodes also have a number of other fields and methods that can be called by the scripts that implements them, and certain subclasses of nodes have unique behaviors. For instance, the MeshInstance class describes a node that will be displayed on the screen.

3.3.3 The Motion of the Player

One of the few pieces of code written in Godot's native language GDScript is the code that controls user interaction with software. It made sense to write this in GDScript as it is a much simpler language than C#, and because a very simple tutorial of first person motion was available in GDScript[13]. However this code required some adaptation to make it work for my purposes, primarily the removal of a number of unnecessary features, and the addition of the ability to move up and down.

The code works using a "head" which is attached to a "neck". Left-right motions of the mouse are translated to left-right rotations of the neck, while up-down motions of the mouse are translated into up-down rotations of the head. keeping the head and neck separate mean that the player cannot turn themselves upside down. Motion is done relative to the head's view, with gentle smoothing applied to motion to give a more natural feeling.

3.3.4 Aesthetic Improvements

I made a number of adjustments to the code in order to make images look clear in my report. I chose a while background opposed to the black of space, and used a low-saturation, simplified image of Earth instead of a more realistic image so that links would show up more clearly over it. For the Earth texture I used a creative commons equirectangular map of the Earth. Equirectangular means that x and y values correspond to longitude and latitude. It was important that the map be equirectangular as this is the format that is compatible with being mapped to a sphere. [14] I also added a soft light source, as this is extremely easy to do in Godot and goes a long way to making the cubes of the satellites appear more clearly and the overall image more pleasant.

I developed a number of different Coloring Methods using the method pattern that allowed me to customize the colors of the various objects in the scene. This was extremely useful as different tests require different ways of highlighting the network to make it clear what I am trying to visualize.

3.4 Initial Observations of the Network

While working on the program I made a few notable observations of the network. In this section I go through each of these observations and outline how it fed into what I was doing.

3.4.1 The Structure of the Network

One of the most notable observations of the network was it's unique, layered grid structure. To highlight this I applied a different hue to each orbit **??**. As you can see, for a given point near the equator, there are in fact two very different kinds of Satellite moving overhead, those going North-East, and those going South-East. Towards the poles, on the other hand, all satellites are moving Eastward. This means that Satellites that pass very close to one another on the equator can end up on opposite sides of the Earth close to the poles. As described below, it is very likely that Satellites will only link to other satellites moving the same direction as them, and in my model I have reflected that. This means

Figure 3.3: The Starlink Constellation with a different hue applied to each orbit.

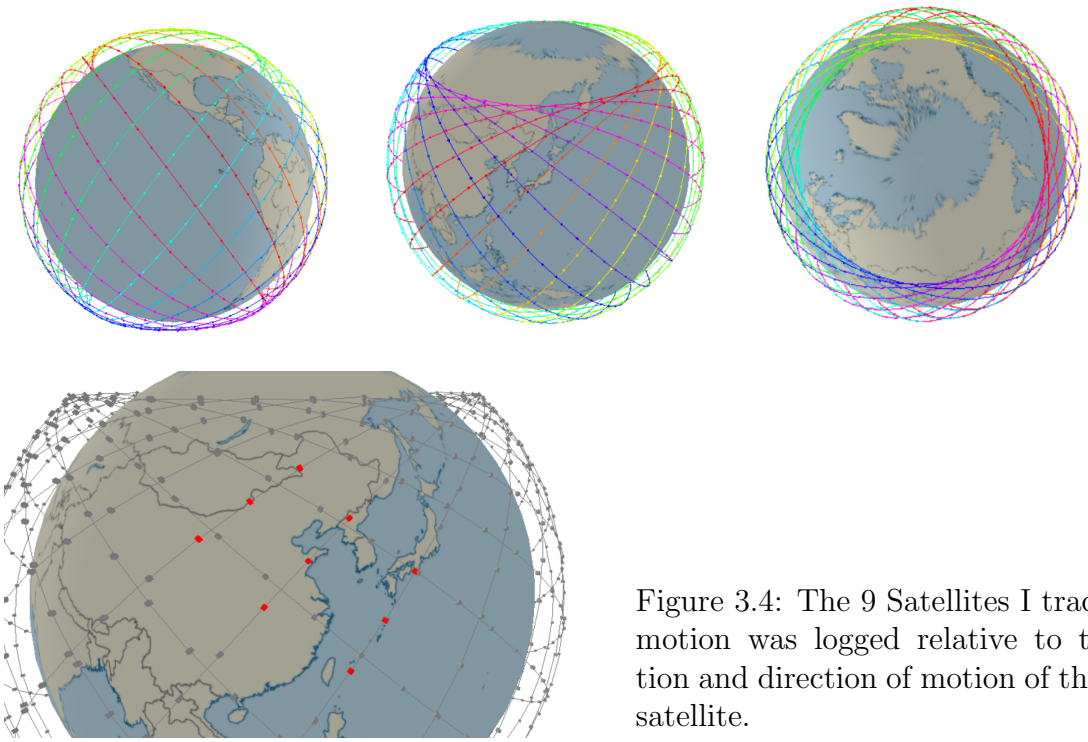


Figure 3.4: The 9 Satellites I tracked, the motion was logged relative to the position and direction of motion of the central satellite.

that for a given point on earth there will be two almost separate grids moving overhead, one grid of satellites moving North-East, and another of satellites moving South-East.

3.4.2 Movement of Satellites Relative to Each Other

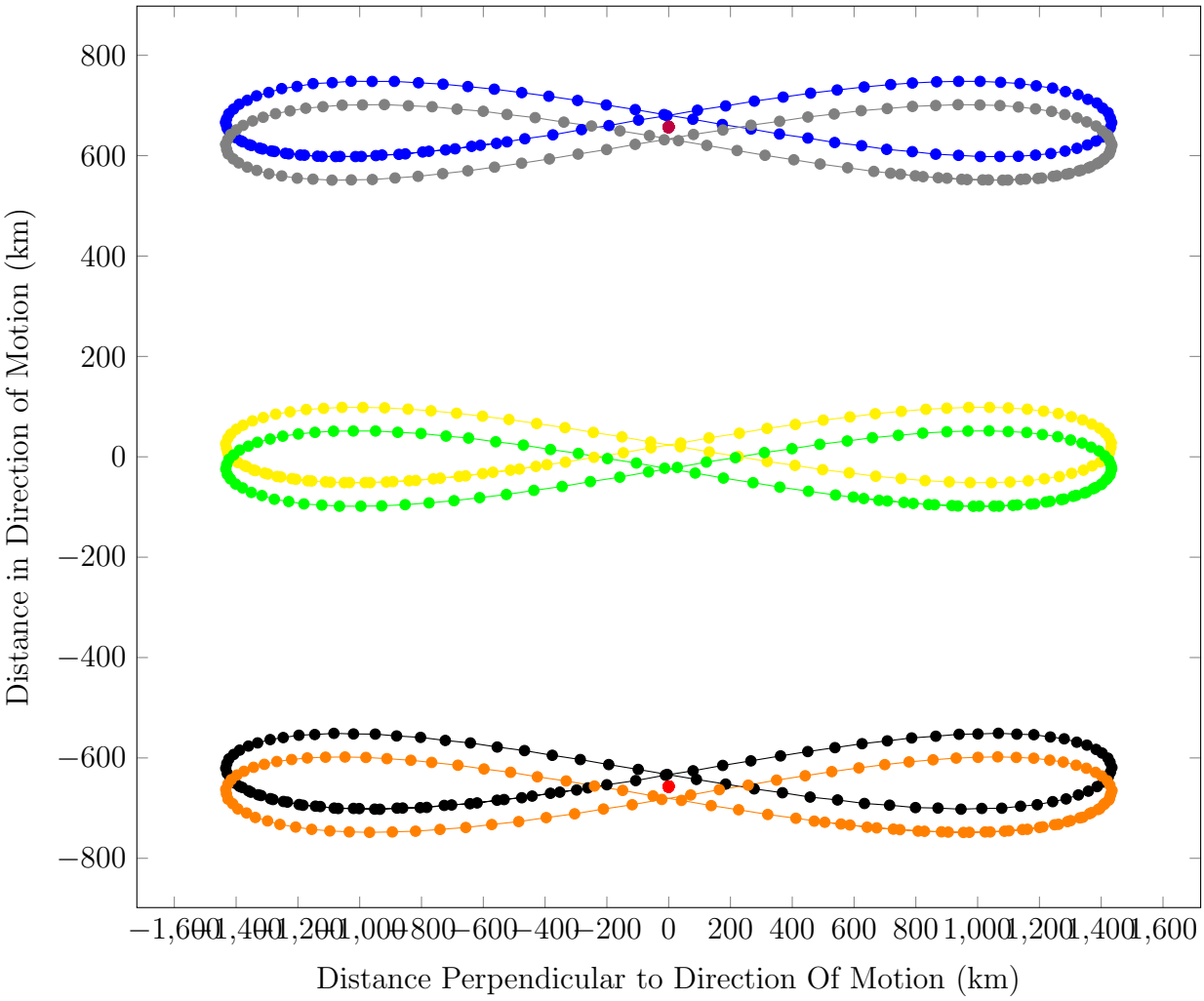
I decided to track the path of a satellite's neighbors relative to it and it's direction of movement over the course of an orbit around the Earth. In figure 3.5 the paths traced by nearby satellites is given. The satellites directly ahead of and behind a given satellite stay in a steady position relative to it and it's direction of motion, while those in adjacent orbits slowly draw figure eights. You can see in figure 3.4 that there are other nearby satellites in different orbits, however these satellites are moving South while the highlighted satellites are moving North, meaning that they will be moving by far too quickly for the main satellite to track them as they pass. [10]

These results show that satellites in adjacent orbits will be moving in relatively slow and predictable ways meaning it should be possible for a satellite to track a laser link to satellites in adjacent orbits. It also shows that linking to satellites in the same orbit will be significantly easier as satellites in the same orbit maintain a constant position relative to one another.

3.4.3 Initial Observations of Latencies

In figure 3.8 you can see an example of the kind of latencies that you might expect between two locations through the SpaceX network. As you can see the latency changes relatively slowly for periods, followed by sudden jumps. The sudden jumps represent when the a new path becomes available or an old path is broken, and a changeover happens, while

Figure 3.5: An example of the paths that neighbors take relative to a satellite and it's direction of motion.



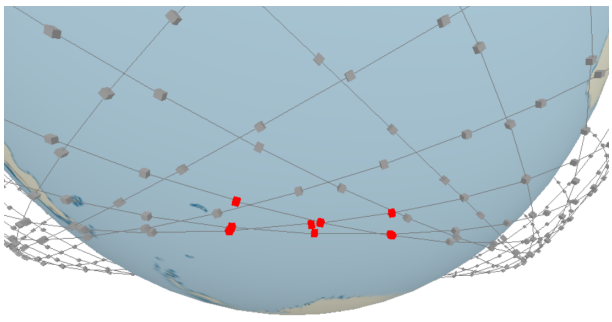


Figure 3.6: Towards the South Pole, the square of satellites fold over and the satellites that were once on the right of the central satellite move to the left.

Figure 3.7: The distance over time from a satellite to it's neighbors perpendicular to it's direction of motion and the direction towards the core of the Earth, or how far to the left or right of the satellite it's neightbors are over time.

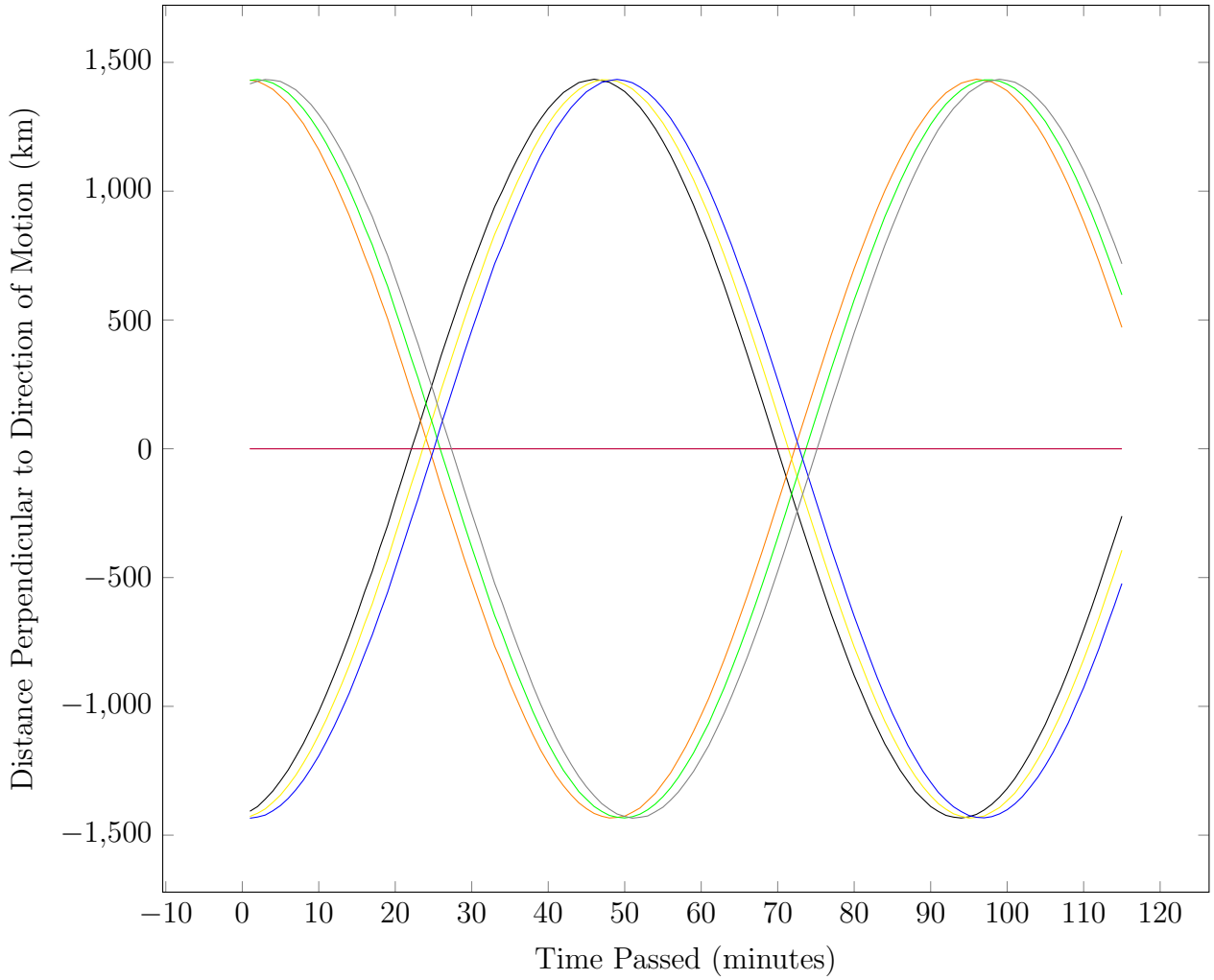
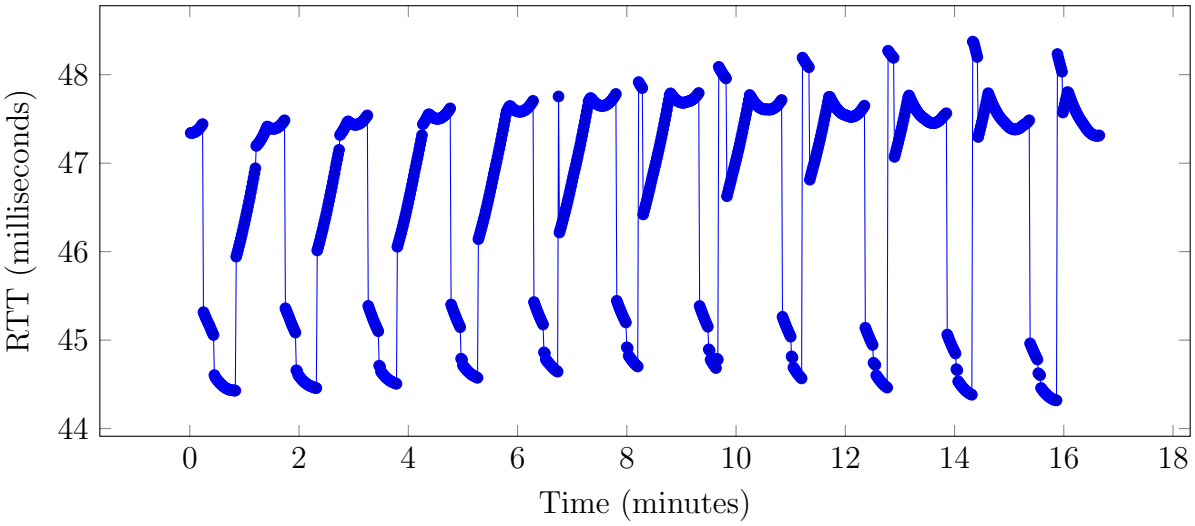


Figure 3.8: The Latency of the London-New York connection over time



the path is constant, the latency changes due to satellites moving away from or towards one another.

In figure 3.9 you can see two different images of the shortest path between London and New York taking at different times. As you can see the shortest path between two locations can change somewhat dramatically over time, this accounts for the huge jumps that are seen in latency.

It's also worth noting that while these jumps are lengthy they are well below the maximum amount of jitter tolerated by VoIP protocols.[16]

3.4.4 Periodicity in Latencies

When initially examining latencies in the network I noticed a slow transition in the result that appeared to suggest some kind of periodicity (figure 3.8). To examine this further, I did a further test, this time running the model for a much longer time (figure 3.10). What I found was a periodicity in the latency data of almost exactly one hour. I connected this to the rotation of the earth under the Starlink network, which has 24 orbits. To confirm this I ran a third test with the Earth frozen, and found the gradual changes observed initially has disappeared (figure 3.11). As far as I can tell, this hour-long variation in the latency of a path is not something that Mark Handley accounted for in his research, however to account for it in mine, all latency results for a link will be determined by taking the average of regular samples across an hour of simulated time.

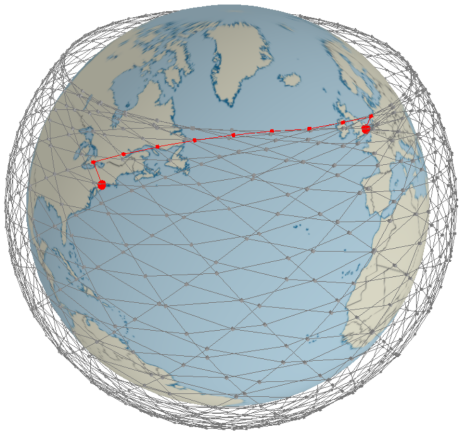
3.5 Selecting my Three Linking Methods

Having got my tests working, my linking operational and having made some preliminary observations I could move on to choosing the three linking methods that I would examine. I chose the following three linking methods.

- $$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = Handley(0)$$

Figure 3.9: Image of the shortest path between London and New York in Handley(-1) at two different times.

(a) The shortest path between London and New York, passing over the artic circle.



(b) The same simulation a few minutes later, now with a path that goes across the south-east moving satellites

Figure 3.10: The Latency of the London-New York connection over time (extended)

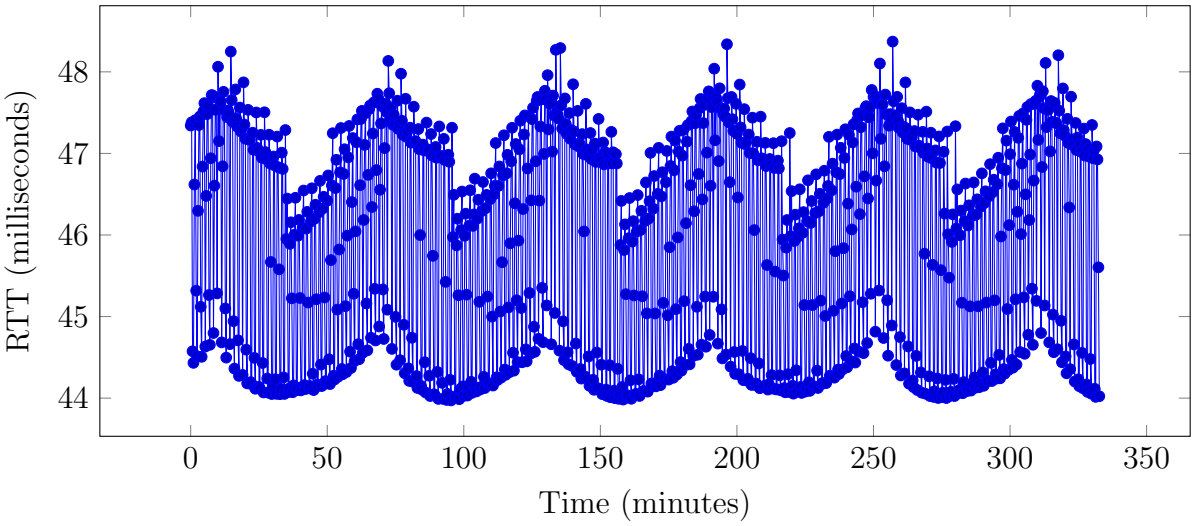
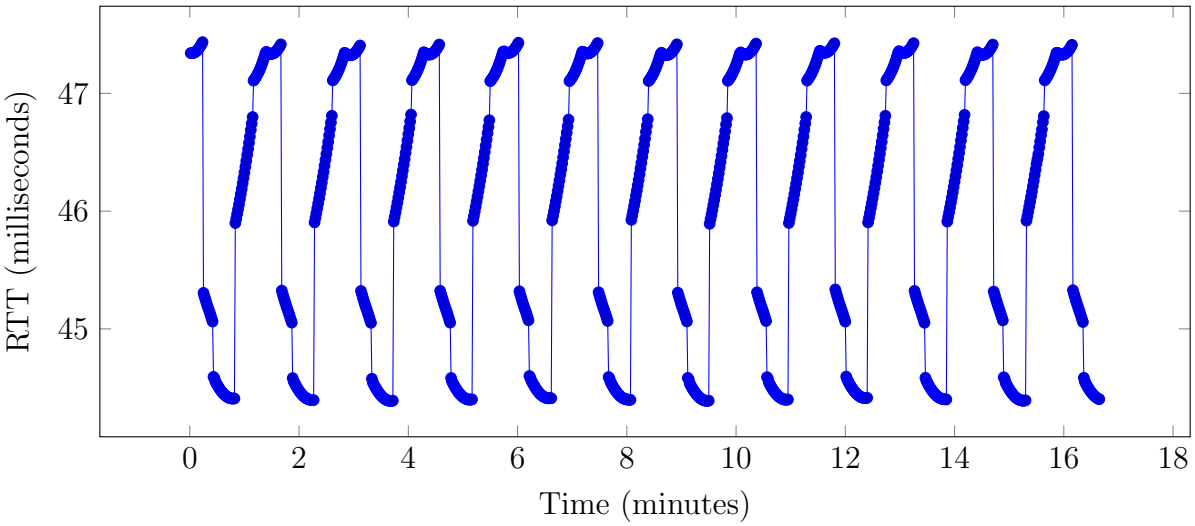


Figure 3.11: The Latency of the London-New York connection over time when the earth is treats as stationary



- $$\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} = Handley(-1)$$
- $$\begin{bmatrix} 2 & 1 \\ -1 & -1 \end{bmatrix} = DistantLinking$$

The first two were relatively obvious picks, they were both examined by Mark Handley and I wanted the opportunity to replicate his results.

DistantLinking came from my observations of the network, I noticed that satellites could comfortably connect with each other over distances much greater than just a single orbit, doing this would reduce the total hop count, and because of the spherical nature of the network, this would quite literally cut corners and reduce the overall distance.

However, there is a problem with this approach. If I chose the linking method:

$$\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

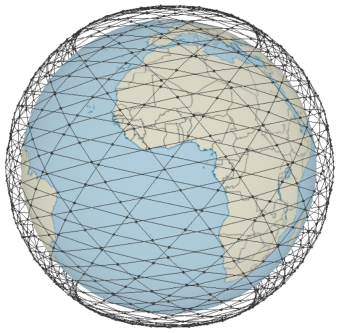
So that each satellite connected to the satellites directly in front of, behind, and two orbits to the left and right, the resultant network would in fact be two completely disconnected components, one network of all the even numbered orbits, and one of all the odd numbered orbits. The network splitting into connected components was common whenever I tried a more complex linking than simply Handley(x). DistantLinking was one of the few linking methods I found that created both long links and remained as one connected component.

In figure 3.12 I show what these linking methods look like in my visualization. In DistantLinking, it is somewhat hard to see what is going on, so I recolored the constellation so that the southward moving satellites were black and the northward moving satellites were white. This can be seen in figure 3.13.

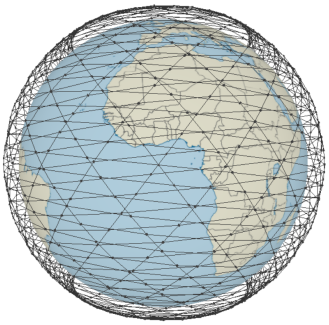
With my three linking methods chosen I could move on to examining their properties and latencies.

Figure 3.12: Images of the three linking methods that I examined

(a) Handley(-1).



(b) Handley(0).



(c) DistantLinking

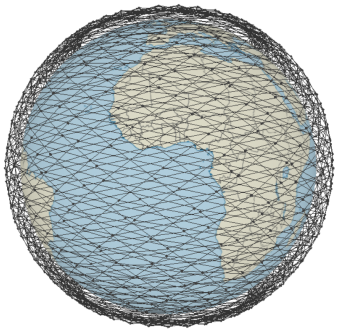
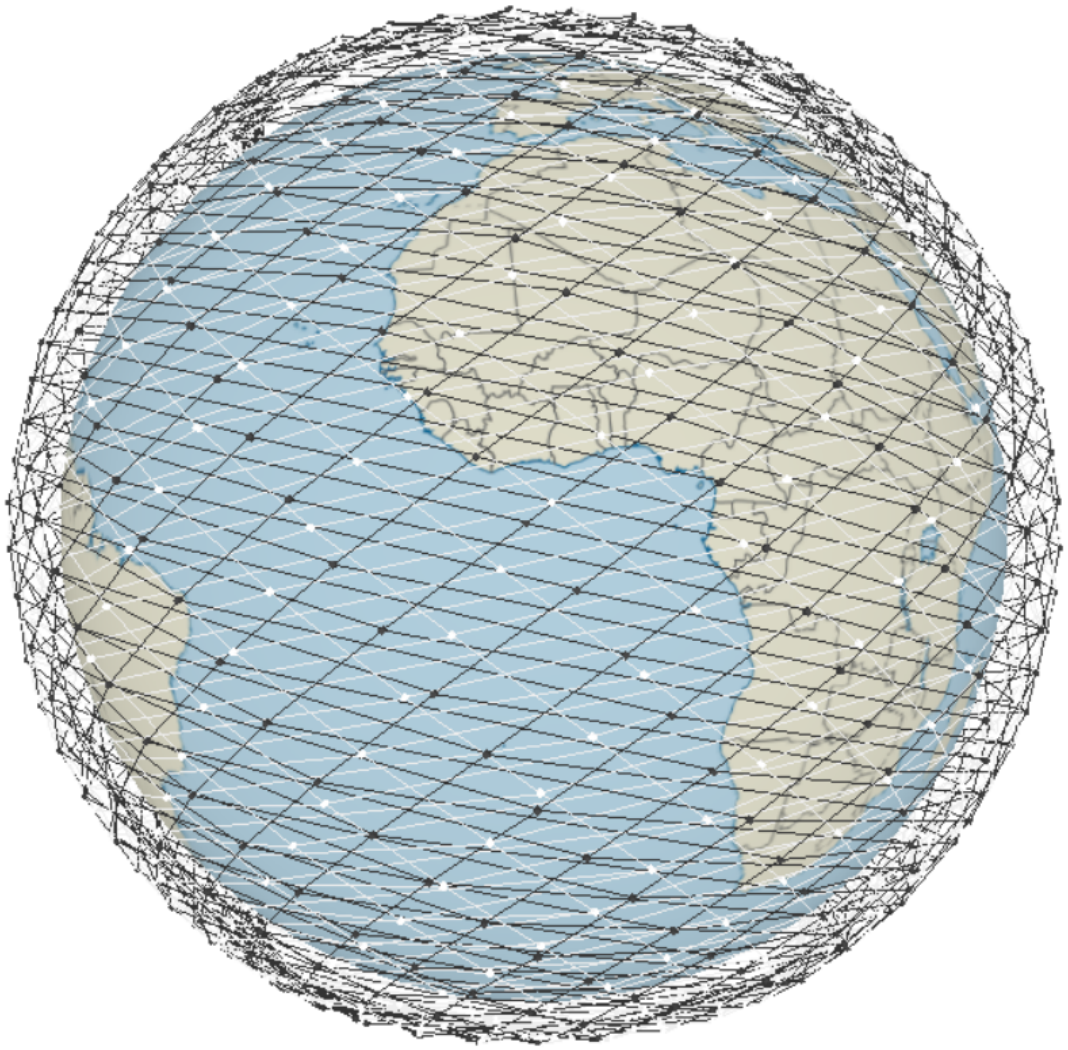


Figure 3.13: DistantLinking with southward moving and northward moving satellites seperated



Chapter 4

Evaluation

My goals going into this project were:

- To develop multiple routing algorithms for this network.
- To compare these algorithms on latency, fault tolerance, and response to high demand.
- To create visualizations of these routing algorithms at work.

With success being determined if I can test three different algorithms, producing meaningful data about their latency, fault tolerance, and response under high demand along with informative visualizations of their operations.

It became apparent early on that "routing algorithms" was a very ambiguous term. It is not clear how one would develop an entirely new routing algorithm to operate on this network, or whether SpaceX would choose to. Furthermore, the behavior of the network is extremely predictable, which means that a base station can easily calculate the optimal path across the network and send a packet up with the exact satellites it needs to be passed between, this makes static routing the most likely option. What is very much undetermined is the topology of the network being routed across, specifically the linking method and the phase offset. Because of that, I chose to examine the properties of 3 linking methods instead.

As stared earlier, the three linking methods I chose to examine are:

- Handley(0)
- Handley(-1)
- DistantLinking

In order to describe these linking methods I have had to create my own notation and methods of describing networks that as far as I can tell from my research have not been created before. Because of that I feel that I have achieved this goal.

4.1 Comparing Linking Methods

4.1.1 Comparing Linking Methods On Latency

As outlined in the figure below 4.5 I took three different paths that I believed tested my linking methods in different ways, London to New York (LDN-NY) was a relatively

Figure 4.1: The RTT of each Linking Method when there have been no satellite failures.

	London - New York	New York - Beijing	London - Johannesburg
Current[3]	75.40ms	198,96ms[1]	163.60ms
Handley(-1)	45.94ms	97.9ms	105.81ms
Handley(0)	46.04ms	97.89ms	134.73ms
DistantLinking	87.47ms	103.4ms	211.37ms

short range link between two cities at similar latitudes. New York to Beijing (BJ - NY) was a very long-range link between similar latitudes, and London - Johannesburg was a link between two very different latitudes. I then tasted the average latency over an hour for all three of my networks. My results can bee seen in figure 4.1. What I found was that Handley(-1) was superior to the other linking methods in all respects, while the DistantLinking was inferior over all three paths. The most notable exception to this was in the very long range NY-BJ path, here all three linking methods had very similar values, I speculate that this is because at an increased distance there is such a wide availability of different paths to choose from that each linking method can choose a path close to the minimum.

The results here are similar to those that Mark Handley was able to achieve in his research, and just like Mark Handley's research, my results show round trip times far shorter than anything currently available terrestrially.

The poor performance of DistL was really surprising to me so I took to the visualization to examine why this was the case. Figure 4.2 shows why this is the case, while short paths of the form I would have hoped did appear occasionally, a great number of times DistL produced paths that were confusing at best.

Another result from my latencies was the relatively poor performance of the London-Johannesburg connection, this is a result that Handley replicated in his research, and is caused by the fact that paths between London and Johannesburg become doglegged, as seen in figure 4.3.

4.1.2 Comparing Linking Methods On Fault Tolerance

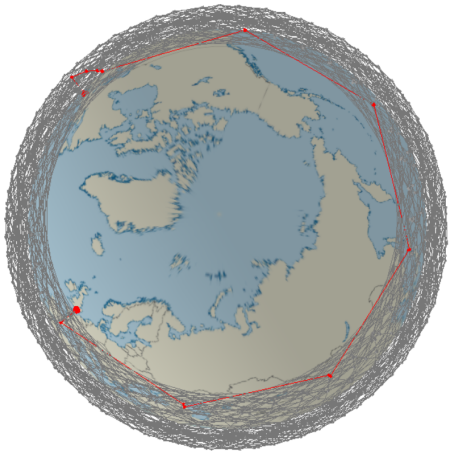
The first test I did for fault tolerance was a simple test of total connectivity. I deleted a certain percentage of satellites from each linking method, and then tested the total number of connected components. I did this 100 times, so that my 95% confidence interval never exceeded 4 components. What I found is that all 3 linking methods had exactly the same curve of connected components, with the expected umber of connected components rising after a 20% failure rate, peaking at an 80% failure rate, and than dropping off as the number of satellites being removed from the constellation outpaced the number of new constellations being crates. This makes intuitive sense, topologically the three linking methods are all grids with the same number of elements, and so it makes sense that they would behave in the same way.

To continue my examination into fault tolerance, I ran 5 hour-long tests of latency for each linking method, path, and proportion of satellites failed (in increments of 1/24ths).

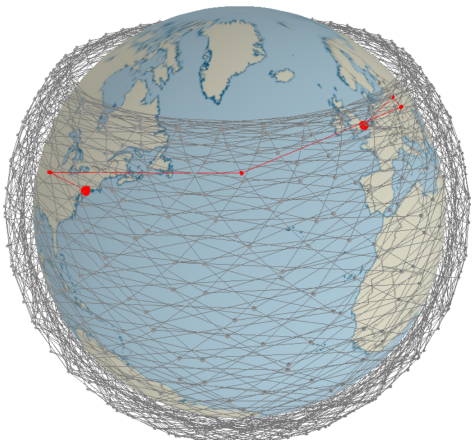
[1]Unfortunately after collecting my data, I discovered that it was very challenging to find the RTT between New York and Beijing, because of this I have had to use the RTT to Seoul instead, which will be comparable but not perfect.

Figure 4.2: Screenshots of DistL creating extremely long links between locations

(a) London - New York: A path routing across the other side of the Earth



(b) London - New York: A short path that appears only occasionally



(c) Beijing - New York: A path with an unusal fold

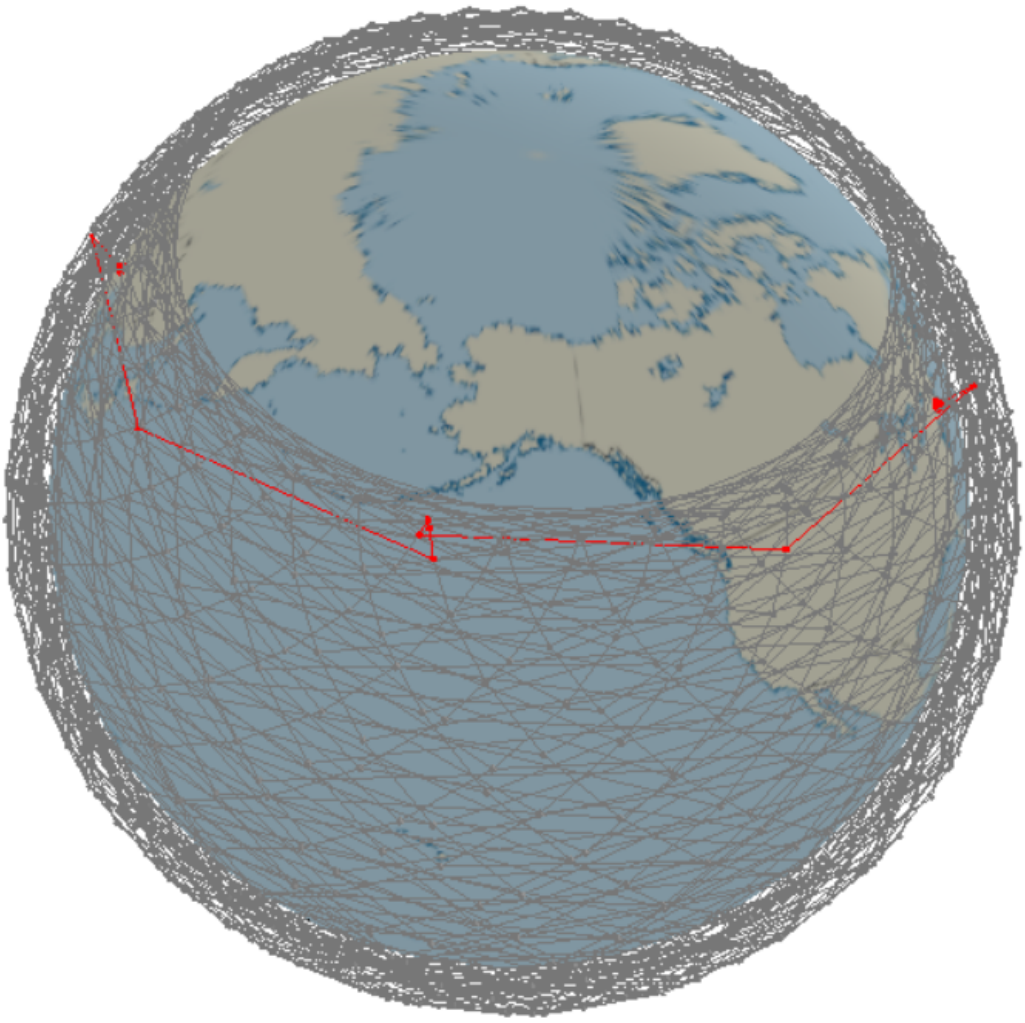
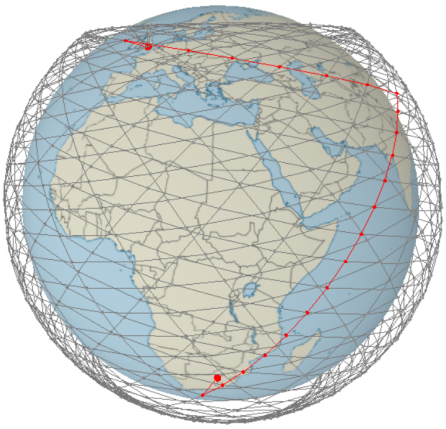
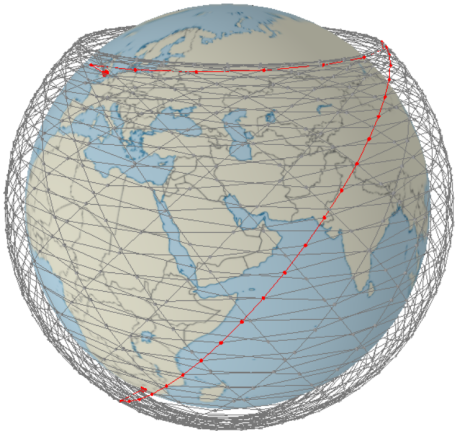


Figure 4.3: London - Johannesburg path under different linking methods

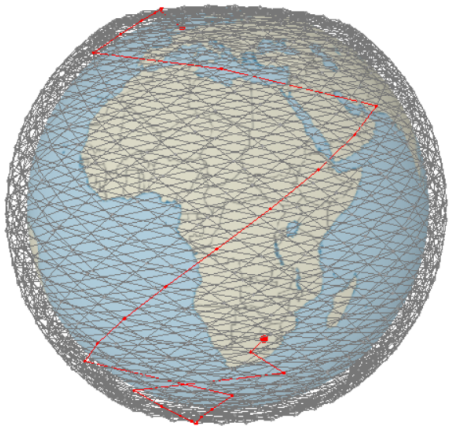
(a) Handley(-1)



(b) Handley(0)



(c) DistantLinking: A very complex path



(d) DstantLinking: A path across the other side of the world

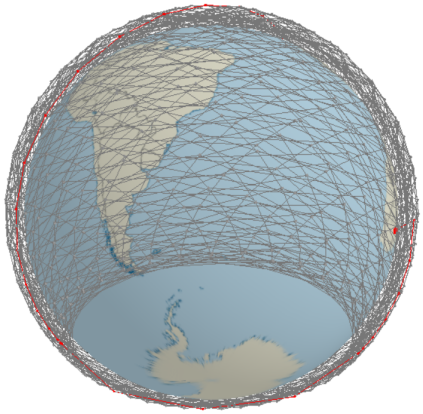
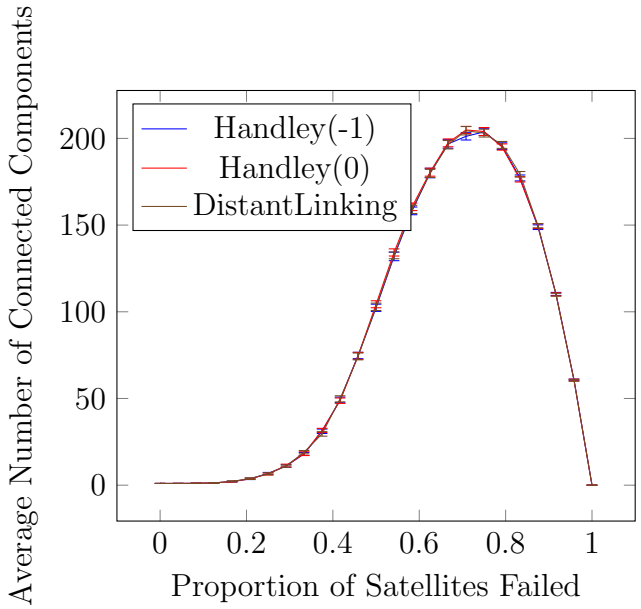


Figure 4.4: The number of connected components after various amounts of deletion of random satellites, using the three linking methods outlined.



I took the average and then used the standard deviation to calculate 95% confidence intervals.

As more satellites failed, there began to be increasing amounts of "downtime" in which a path could not be established between the two ground stations. I have ignored these values when calculating path latency, but I have reflected this by including keeping track of the proportion of time that a path is available.

There are a huge number of interesting observations that can be made from this data.

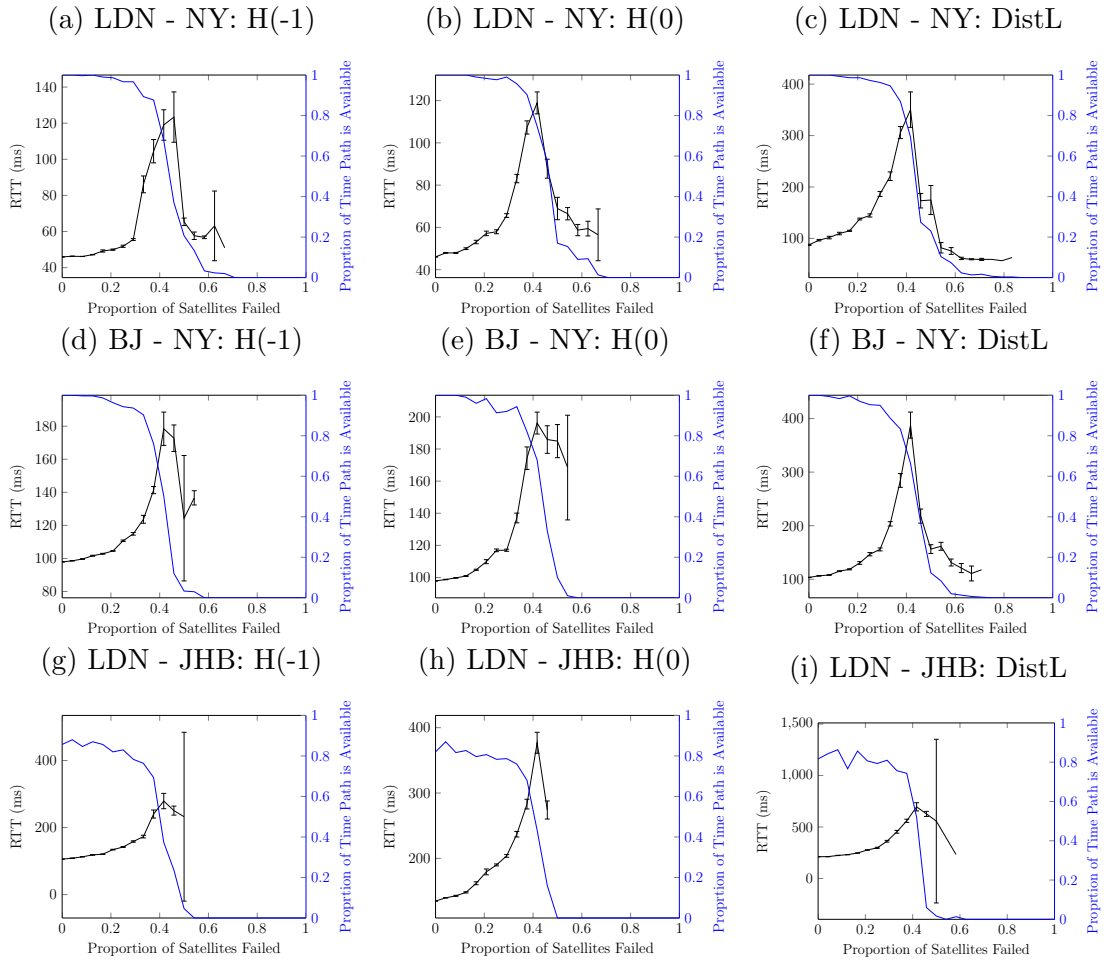
First, all three linking methods and all 3 routes have a similar general behavior. As the failure rate of satellites increases, the RTT steadily increases, while the proportion of time that the link is available steadily falls. When, at around 45% (where the average number of connected components is increasing fastest) there is a sudden drop off in the proportion of time that a link is available. After this point, for a majority of the time, there is no link between the two cities. When there is, however, this link is significantly shorter. This doesn't mean that the network has spontaneously gotten better at routing with less satellites, it just means that a steady connection between the two cities is so rare that on the few occasions it is formed, it ends to be more simple.

Another interesting observation that can be made is that the LDN-JHB link is never at 100% availability. This is because Johannesburg is at a latitude such that between certain times of the hour there is no satellite above is that can be connected to even without any failures. This is a trait shared by all cities closer to the equator, and poses a significant limitation on the SpaceX network.

As to be expected, LDN-NY, which is a shorter path, was generally more fault tolerant than the longer BJ-NY and LDN-JHB paths. However, as seen in 4.6 there is no difference between the three linking methods when it comes to how they respond to failures. All this together suggests that there is no static linking method that will be any more resistant to faults in the network as these are.

All together, these results suggest the critical value for Starlink, a failure rate after which the network will be unable to function properly, is nearly 40%. This is a very high

Figure 4.5: The RTT and up time of various paths under each linking method.



value. Starlink claims it's satellites will have a lifespan of many years, which means that such a failure rate is extremely unlikely to occur.

Some images of the staggered paths and disconnected networks formed at various failure rates can be seen in Figure 4.7.

4.1.3 Comparing Linking Methods On High Demand

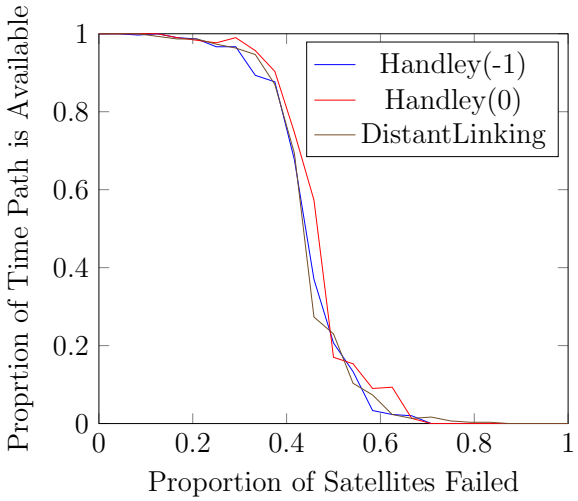
I did not have time to calculate n shortest paths around the network, though I believe that some knowledge about how the Linking Methods might respond to high demand can be gleaned from how they respond to satellite failure, as the ability to resist failure is connected to having multiple different ways to route between two points, which in turn is connected closely to fault tolerance. Because of this, I would expect all three linking methods to be equally effective at dealing with high demands.

4.2 Visualizing the Linking Methods at Work

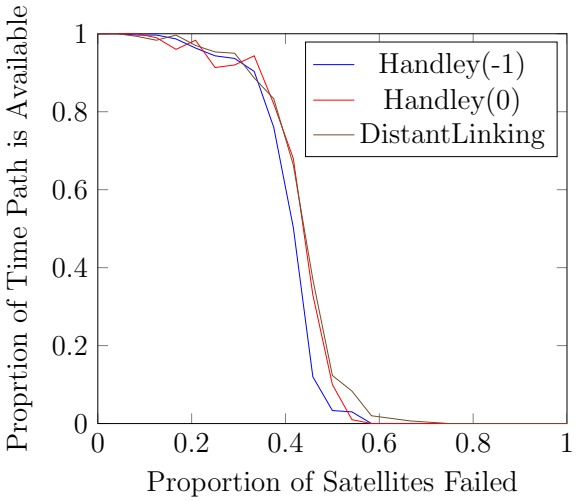
I believe that my goal of visualizing the linking methods at work has been effectively achieved through the available executable and screenshots that have been taken. By visualizing the network like this I believe that I have gained, and have the ability to give to others, and intuitive understanding for how this network works that can direct

Figure 4.6: The proportion of time that a link is available vs the proportion of the satellites that have failed for 3 links.

(a) London-New York.



(b) New York-Beijing.



(c) London-Johannesburg.

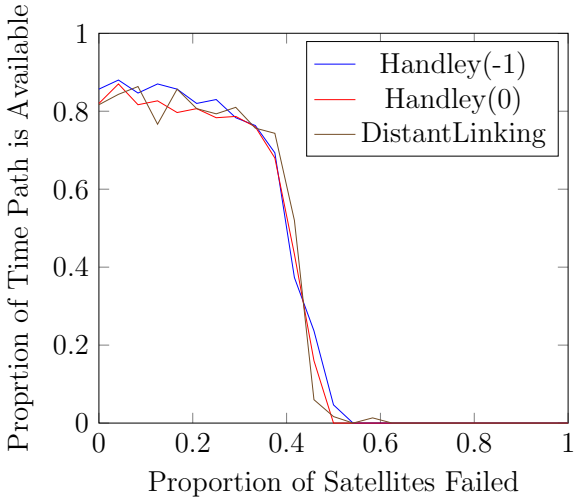
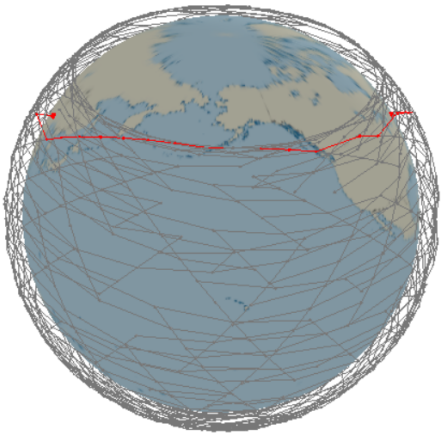


Figure 4.7: Screenshots of Various Linking Methods at Different Failure Rates Responding to the failure rate

(a) NY-BJ, 70% faliure rate

(b) NY-BJ, H(0), 17% faliure rate



(c) LDN-JHB, H(0), 21% faliure rate

further research into it's properties, as well has having contributed to and directed my own research into it's properties.

4.3 Direction and Focus

Maintaining a consistent goal has been challenging during such an open-ended project. If I were to do this project again I would more carefully choose my objectives I believe that I made a mistake in talking about "routing algorithms" in my original proposal, as while linking methods could be considered a form of routing algorithm, these concepts are fundamentally different, and I should not have

A number of projects in the early phrases had to be abandoned not just because of inefficiency but because new information came about that rendered them useless.

The announcement from SpaceX that they were significantly changing the constellation had massive impacts to the project. Firstly, the new satellites would have only four links instead of five, this implied that the satellites would not not have a "free" link but would instead be connected only to those ahead of and behind them. Because of this, I decided to drop the free links from my model, but doing so meant that a large chunk of code that had been written was discarded.

The knowledge that SpaceX were willing to change their designs so abruptly also had a significant impact on modeling their constellation. It is likely that the second and third phases of Starlink will be similarly changed to be in similar altitudes to phase one, however without a full understanding on what these new spheres would be, or how they would connect to phase one, it was elected to instead focus solely on the first sphere's constellation.

Overall I think that this project has been somewhat of a success, in that I have contributed in a meaningful way to a growing body of research and confirmed and supported the finding of the researchers at UCL. But because of the poor initial planning of the project I cannot call it a full success.

4.4 Using Godot

There were numerous issues caused by using Godot for this program. Unfortunately the C# features in Godot are still under development and have proven to be very buggy, with numerous crashes requiring altering source files in text editors and crashed that would corrupt my .mono repository and render my entire program unusable. While open-source does provide benefits in allowing me to share my code with full confidence, I believe that this project would have been improved if programmed in a more refined engine such as Unity.

Chapter 5

Conclusion

I have shown, through examining three different linking methods, that we can expect similar fault tolerance and response to high load from any grid-like linking method that SpaceX chooses to employ, but that these linking methods vary significantly in the expected latencies. I also believe that I have shown that the linking method that SpaceX choose to do with will most likely be of the Handley(x) form, rather than anything more complex.

I also believe that I have in the process of writing this report made a number of other discoveries that will inform future research. The discovery of the 60-minute variations in latencies is in my opinion very significant, this is something that I have not seen observed in any of the research thus far but has a significant impact on how one calculates latencies across the network. The discovery that locations close the equator will not be consistently connected to the Starlink network is also very notable.

5.1 Extension

Were I to come back to this project I would do a number of things differently. I think that I have been let down by a lack of imagination in how linking methods might work, all of the linking methods I ended up choosing where fundamentally grids and therefore limited in the differences in their behavior. I would have liked to have experimented with more complex linking methods, perhaps ones that divided up satellites into two classes, one for long ranged communication and one for short ranged, creating a condensed grid. I wild also have liked to experiment with allowing satellites to respond to failures in the network, for instance by moving into a gap left behind by a failed satellite or by connecting to the satellite on he other side of the gap.

I also think that it would have been worthwhile to examine how such a network would connect to other orbital spheres. SpaceX intends to send up more spheres of satellites in the future, but how / if these spheres of satellites will interact is as of yet unknown. Modeling various ways that two spheres of satellites could communicate, especially when both spheres have only four links per satellite, would be incredibly interesting.

Building off of the data collected on the changes in path latency, I think simulations of different implementations of TCP under these kinds of latencies would be very worthwhile, to see if the variation in latencies is tolerable by TCP. And I think it would be worthwhile to simulate how TCP behaves around the sudden changes in latency created when a new path is opened up, while these changes no not happen frequently, TCP is very sensitive to packet reordering and I would be curious to know if it can handle the packet reordering

caused by such a significant change in latency and if there are any protocols for path hand over that would improve this. As far as I know no such experiments have yet been done, and it is an opportunity to build off of my current work that I think would be really valuable.

Bibliography

[1] Daniel Pflughoeft. High Speed Priority Queue for C Sharp. 2013. Available from: https://github.com/BlueRaja/High-Speed-Priority-Queue-for-C-Sharp

[2] Dr. James McCaffrey. Priority Queues with C#. Visual Studio Magazine. 2012. Available from: https://visualstudiomagazine.com/Articles/2012/11/01/Priority-Queues-with-C.aspx

[3] WonderNetwork. Global Pink Statistics. Available from: https://wondernetwork.com/pings

[4] Exede. Website. Available from:https://www.exede.com

[5] Hughes. Website. Available from: https://www.hughes.com

[6] Elon Musk. Twitter. 2018. Available from: https://twitter.com/elonmusk/status/1000453321121923072

[7] Elon Musk. Twitter. 2019. Available from: https://twitter.com/elonmusk/status/1127388838362378241

[8] Federal Communications Commission (FCC). File Number = SAT-LOA2016111500118. 2018. Available from: licensing.fcc.gov/cgi-bin/ws.exe/prod/ib/forms/reports/related_filing.hts?f_key=-289550&f_number=SATLOA2016111500118

[9] SpaceX. Attatchment A: Technical Informaton to Supplement Schedule S. Federal Communications Commission (FCC). 2016. Available from: https://licensing.fcc.gov/myibfs/download.do?attachment_key=1158350

[10] Mark Handley. Delay is Not an Option: Low Latency Routing in Space. 2018. University College London. Avaialble from: http://nrg.cs.ucl.ac.uk/mjh/starlink/

[11] Hughes. Hughes High-Throughput Satellite Successfully Launched, Setting the Stage for the Next Generation of Satellite Internet. 2016. Available From: https://www.hughes.com/who-we-are/resources/press-releases/hughes-high-throughput-satellite-successfully-launched-setting?locale=en

[12] Gunter's Space Page. Jupiter 2 / EchoStar19. 2015. Available from: https://space.skyrocket.de/doc_sdat/jupiter-2.htm

[13] Turtletooth. Godot First Person Controller. 2017. Available from: https://github.com/turtletooth/GodotFirstPersonController

[14] Wikipedia. USGS majplatecolor.png. 2012.Available from: https://commons.wikimedia.org/wiki/File:USGS_majplatecolor.png

[15] C.D. Murral and S.F. Dermott. Solar System Dymanics. 1999. Cambridge University Press.

[16] VoIP Studio, VoIP And How Much Jitter Is Acceptable? 2015, Available from: https://voipstudio.com/blog/voip-how-much-jitter-is-acceptable/

[17] Chris Davis, "As launch nears, SpaceX's Starlink satellite internet gets big change approval" SlashGear, 2019, Available from: https://www.slashgear.com/as-launch-nearsspacexs-starlink-satellite-internet-gets-big-change-approval-29574

Proposal Final.pdf

Modelling and Visualisation of Networking in Low Earth Orbit Constellations

Project Proposal

Introduction

SpaceX are planning to launch a constellation of 4,425 low Earth orbit communication satellites in the next few years. The objective of this constellation, called Starlink, is to provide low-latency internet connection across the world.

The satellites in this network will be in constant motion, not just relative to the ground, but relative to one another, creating a network with a constantly changing topology and associated latencies. The question of how to route signals across such a network has not been thoroughly explored, but it will become increasingly relevant as more and more companies build similar constellations.

My goals are:

1. To develop multiple routing algorithms for this network.
2. To compare these algorithms on latency, fault tolerance, and response to high demand.
3. To create visualisations of these routing algorithms at work.

In order to achieve goals (1) and (2) I will have to do extensive research into routing algorithms, and then translate the strategies I find into an unfamiliar context. To do (2) I will need to create a model of SpaceX's constellation of satellites, and for (3) that model will have to be visualised. To achieve this I will be using a games engine to create a simulation of SpaceX's constellation. While this will sacrifice some precision, a precise model of so many intersecting orbits at different altitudes and the connections between them is exteremely difficult, and a games engine will allow me to create visualisations that can be ran on any device.

The visualisation should be able to take any two points on the Earth and show the path of a data packet between them, as well as showing the network under high demand, under different routing algorithms. The results produced can then be used in (2). In order to achieve (3), my model will have to be clear and well designed, and I will use design principles in its construction.

Ultimately, all three of my goals shall feed into one another, as (2) and (3) open up new insights into the network, that can help inspire (1).

Further Details About SpaceX's Network

As my example constellation I will be using the constellation planned by SpaceX. Theoretically, I could have used any constellation, and focusing my attention on SpaceX's runs the risk of creating algorithms that are specific to it, which might not perform as well on other networks. However, the vast amount of legislation on satellites makes it hard to know what a "normal" network would look like. By using SpaceX's constellation we get a confirmed legal and physically possible network, on which we can build our routing algorithms.

There is a lot we do not know about this constellation, but this is what we can infer from SpaceX's application to the FCC[1]:

- 4,425 satellites will orbit in 83 different orbital planes at altitudes ranging from 1,150km to 1320km. We know exactly how many satellites will be in each plane.
- For our purposes these satellites are identical.
- Connections will be made with lasers, which require line-of-sight.
- Base stations can only communicate with satellites at an elevation degree of at least 40 degrees.
- One base station can receive signals from 4 different satellites simultaneously without interference, each satellite can send two signals to a base station simultaneously. Which means that a base station can receive 8 total signals at once.
- The satellites have an expected lifespan of 5-7 years.

Criteria for Success

This project will be deemed a success if I can test three different algorithms, producing meaningful data about their latency, fault tolerance, and response under high demand along with informative visualisations of their operations.

Starting Point

The initial inspiration for this project comes from a talk given my Mark Handley on Sep 25th, 2018, however I will not be using any of the code that he used in his talk. My primary reference point will be SpaceX's reference point to the FCC, though this will only be used for deciding on variables in my model. In principle, my model should be applicable to any constellation of satellites.

When I am developing algorithms, I will research a wide variety of texts on routing algorithms. As far as I know, there are no public algorithms for a network with these properties, and if I encounter one in my research I will endeavour to create my own.

---

[1] licensing.fcc.gov/cgi-bin/ws.exe/prod/ib/forms/reports/related_filing.hts?f_key=-289550&f_number=SATLOA2016111500118

Timeline

19th Oct	Proposal Submitted	
2nd Nov	Satellites Modelled	By this point I should be able to model the orbit of my satellites around the Earth, and know when two satellites are able to communicate to one other or to a base station.
16th Nov	Connections Modelled	At this point I should be able to efficiently model and visualise a connection between two satellites, or between a satellite and base station.
30th Nov	Data Flow Modelled	At this point I should be able to efficiently model and visualise a data packet moving between two base stations through the network through a given path.
14th Dec	Routing on a Static Network	By now I hope to have designed a simple test algorithm, working on static satellites, that can be visualised in operation.
28th Dec	Random Routing on a Moving Network	Now, by adding back in orbits, I should be able to visualise a random routing algorithm operating on this moving network.
11th Jan	Tests Designed	By this point I should have a series of pre-programmed tests to apply to a routing algorithm. I can then use these on my static and moving cases to demonstrate they are effective.
25th Jan	Iterate on Visualisations	Part of the success of my project relies on my visualisations being clear and informative. To ensure I achieve this, I'm going to dedicate some time to reviewing and iterating on my visualisation.
1st Feb	PROGRESS REPORT DUE	
8th Feb	Selection of Algorithms	At this point I should have completed three proposals for algorithms to test.
22nd Feb	Algorithms Written in	I will design each algorithm individually. One algorithm will most likely be much more involved than the others,

		and writing that algorithm will consume the bulk of my time.
8th Mar	Algorithms Tested	By now my algorithms will have been fully tested and data will have been produced.
22nd Mar	Iterate on Visualisations	Once again I will iterate on my visualisations, considering design principles and asking if my visualisations are still clear and informative when applied to my more advanced algorithms.
5th Apr	Model and Algorithms Complete	By this point my model should be fully functional. These two weeks are dedicated to bug testing.
26th Apr	PROJECT WRITTEN	At this point I the code of my project should have been completed and all the relevant data should have been collected, and my dissertation should have been fully written. The next two weeks will be dedicated to publishing and editing my completed dissertation.
10th May	PROJECT COMPLETE	This date, a week before the deadline, is when I plan to have sent my dissertation to my overseers.
17th May	FINAL DEADLINE	

Resources Declaration

For the modelling of this system I will be using the games engine Godot on my personal laptop. Using a games engine will allow me to create real-time 3D models of the constellation in motion, and Godot is free, open-source, intuitive and well-supported. Godot has two supported languages, it's native language GDScript and C#. GDScript, which is weakly typed, will be used for the visualisation, while the strongly-typed C# will be used for the bulk of the simulation, where precision is more important.

I will be using my own laptop. To mitigate the risks of this, I will back up my code to github every day.