Interactive Visualisation System for Wide Area Network Paths

Authors

Ibrahim Al-Qaysi <iaq@kth.se>

Farzad Hajibagheri <farzadha@kth.se>

Emmanuel Chaudron <chaudron@kth.se>

Wei Wang <wew@kth.se>

Biwen Zhu <biwen@kth.se>

Abstract

Cloud services are used by users on daily basis. These services are latency-sensitive and the quality of service is negatively perceived by the users due to latency. This issue can affect the user experience and the revenue of the service provider. The project will investigate how to reduce the network latency which is affected by network topology in wide area networks and improve the performance of geo-distributed systems. The Latency is proportional to the distance between network hops. By reducing this distance we are reducing the latency in wide area networks. This project requires knowledge in SQL language for interpreting, understanding and analyzing the database content. This will enable us to make plots of information found in the database tables and to extend the database by creating new tables that will contains geographical information. Unfortunately, a problem has occurred while extending the database. The provided dataset generated by the traceroute led to few hops instead of multiple hops. The provided database has to be ignored for entirety of the project. An important aspect of this project is to create and develop an Interactive Visualisation System (IVS), through which network topologies are going to be displayed on the world map. The IVS contains some GUI elements and different network topologies to choose from. The last step in this project is to design and develop Expert Infrastructure Change System (EICS). This system is capable of proposing an alternative connection(s) on the graph based on the available information and available resources to reduce the network latency. This system will help to find the shortest path in the network topology so that the average distance between two nodes is reduced.

Keys words

Network latency, Data analysis, Interactive Visualisation System (IVS), Expert Infrastructure Change System *(EICS).*

Contents

1. Introduction

Cloud services are being used by wide range of users on a daily basis. These services are being replicated and deployed on top of other cloud services across geo-distributed datacenters. Most of these services are latency-sensitive and the quality of service (QoS) is negatively correlated as perceived by the users due to latency. This can directly affect both the user experience and the revenue of service provider.

1.1 Problem statement

This project will investigate how to reduce the propagation delay in wide area networks (WAN) and improve the performance of geo-distributed systems. If the best network path within the existing network topology can be calculated and proposed by an algorithm then the latency can be decreased and the QoS of a geographically distributed system can be improved.

1.2 Problem

Network latency is highly affected by network topology in WAN(s). How statistical analysis of network paths may help us to improve network topology?

1.3 Hypothesis

When using existing topologies we can propose a new topology that can reduce network latency and is it possible to design a system to improve the existing network topologies using offline data to improve current network topology.

1.4 Purpose

The purpose of this project is design and develop an expert system with an interactive visualization system that can be used to reduce the latency in wide area networks and display the result on the world map. These systems can help companies no matter the size to reduce network latency between their datacenters. By reducing the latency in the network, companies can provide better services to their customers and improve their profitability.

1.5 Goals

This project has three goals:

1. Interact with MySQL database and interpret network path information to analyse a dataset containing network path measurements and identify statistical properties of these network paths.

2. Design and develop an Interactive Visualisation System (IVS) that combines individual path measurements into a graph and project this graph onto the world map in an interactive fashion.
3. Design and develop an Expert Infrastructure Change System (EICS), Integrate this system IVS before evaluating the complexity of this EICS algorithm and its effectiveness.

1.6 Methods

A combination of qualitative and quantitative research were chosen because they were considered best suited for this project. The literature study provided enough knowledge and materials for this project. The visualization tools that were chosen from the literature study were evaluated according to the project evaluation criterion. The implementation required programming language for different programming languages. The evaluation was made to help the project to choose a suitable visualization tool for this project.
A more detailed description of the research methodology can be found in Section 3.

1.7 Measurable objectives

An EICS integrate with IVS, which capable of proposing alternative connection(s) on the graph based on the available information and resources. Once it found a solution, it renders new vertex/vertices with edges in the network topology on the world map.

1.8 Evaluation Techniques

For the algorithm, making different changes based on an existing network topology, it can always show the correct information of all the network paths between the source and destination hops and find the best network path.

Examining the IVS using different network topologies to display the best path that gives us the shortest path from source node to destination node. Google Maps has been chosen to do network visualization since it has rich mapping features.

1.9 Ethics and sustainability

This project is about reducing network latency for companies that are in a constant battle with it. The project does not create any direct ethical dilemma, but what the result can be used for can create one. We believe the result can benefit companies no matter the size and society as it improves quality of service perceived by users and companies' profitability.

2. Extended background

This section contains a detailed background about the project terms, concepts and tools that have been used throughout the project work. The section also provides a more in-depth understanding of the relevance of these terms and concepts for the work. The first part of this section deals with database that has been used for the project. Most of the work on databases required a virtual machine to be installed and configured with a remote virtual machine that has been provided for this project. Furthermore, we describe in details about visualisation tools that have been considered to use in the project, and the terms that we considered important for the read to understand such as latency, RTT and traceroute. Moreover, we describe the Dijkstra's algorithm that will be used together with the expert system.

2.1 MySql database

The database that has been provided for the project contains information that needs to be analyzed. These information are found in tables and have name identifiers, id(s) and values that can differentiate them from each other. The tables are connected with each other via primary keys and foreign keys. The figure below shows how the database tables are connected and the relationship between them.

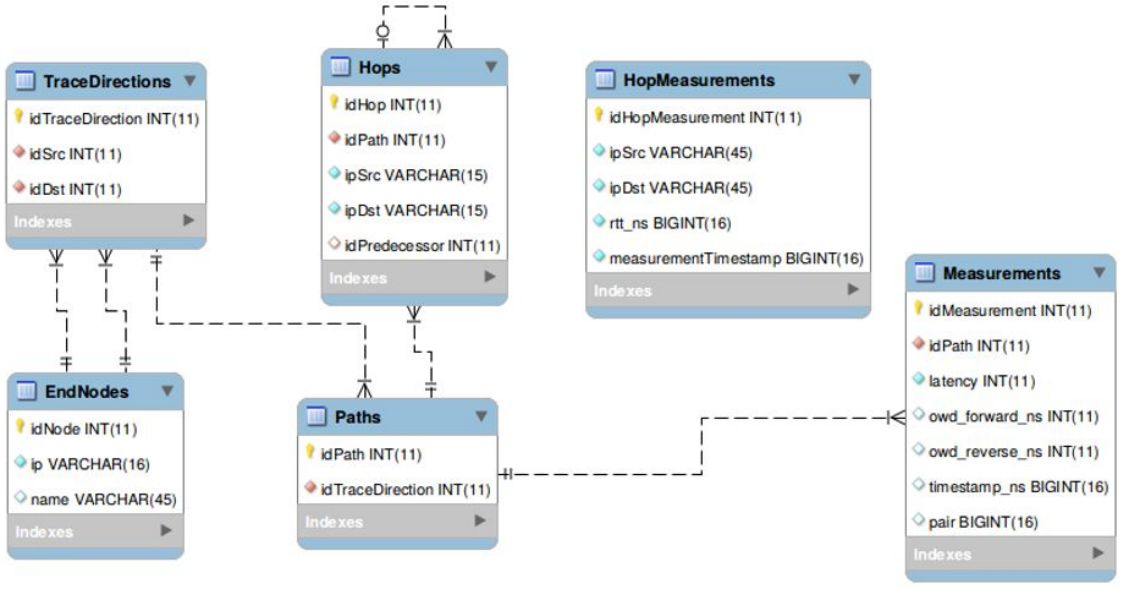


Figure 1: Database tables and their content that are being used for this project [1].

Short Description of each table.

EndNodes Table

This table contains three columns idNode, ip, and name. The idNode contains a unique identifier for every datacenter in the table. The ip column contains ip addresses for the datacenters. The last column is name which contain the names of the five datacenters located in Ireland, Tokyo, Oregon, Seoul, N.Virginia.

idNode	ip	name
1	34.253.136.165	Ireland
2	13.113.103.115	Tokyo
3	34.211.167.25	Oregon
4	13.124.159.69	Seoul
5	34.229.118.49	N.Virginia

Figure 2: first 5 lines of EndNodes table

Field	Type	Null	Key	Default	Extra
locId	int(10) unsigned	NO	PRI	NULL	
country	char(2)	NO		NULL	
region	char(2)	NO		NULL	
city	varchar(64)	NO		NULL	
postalCode	varchar(5)	NO		NULL	
latitude	float	YES		NULL	
longitude	float	YES		NULL	

Figure 3: Structure of EndNodes table

GeoIp Table

This table shows the mapping relationship of locId and range of IpNum (between startIp and IpNum)To calculate the decimal address from a dotted string, perform the following calculation: (first octet * 256^3) + (second octet * 256^2) + (third octet * 256) + (fourth octet). Conservely ,we could change these ip num to ip address instead.

Field	Type	Null	Key	Default	Extra
startIpNum	int(10) unsigned	NO	PRI	NULL	
endIpNum	int(10) unsigned	NO	PRI	NULL	
locId	int(10) unsigned	NO		NULL	

Figure 4: Structure of GeoIp table

startIpNum	endIpNum	locId
0	0	0
16777216	16777471	609013
16777472	16778239	104084
16778240	16779263	17
16779264	16781311	47667

Figure 4: first 5 line of the GeoIp table

HopMeasurements table

This table shows the mapping relationship of each hop's idMeasurement and ipSrc and ipDst, as while as the measurementTimestamp.It also shows the rtt (round-trip time (RTT) is the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgment of that signal to be received). Moreover, it shows the flow direction of each hop, (ipsrc and ipdst pairs).

Field	Type	Null	Key	Default	Extra
idHopMeasurement	int(11)	NO	PRI	NULL	auto_increment
ipSrc	varchar(45)	NO		NULL	
ipDst	varchar(45)	NO		NULL	
rtt_ns	bigint(16)	NO		NULL	
measurementTimestamp	bigint(16)	NO		NULL	

Figure 5: **the Structure of table HopMeasurements**

idHopMeasurement	ipSrc	ipDst	rtt_ns	measurementTimestamp
1	172.31.22.229	100.65.11.161	977203	1499538024002969774
2	172.31.22.229	176.32.107.12	225852339	1499538024002969774
3	172.31.22.229	52.93.7.52	223554643	1499538024002969774
4	172.31.22.229	52.93.6.149	226128930	1499538024002969774
5	172.31.22.229	54.239.53.9	222918012	1499538024002969774
6	172.31.22.229	54.239.53.72	219772901	1499538024002969774
7	172.31.22.229	54.239.53.85	221283258	1499538024002969774
8	172.31.22.229	54.239.53.36	223291926	1499538024002969774
9	172.31.22.229	54.239.52.141	223555747	1499538024002969774

Figure 6: **first 9 rows of table HopMeasurements**

Hops table

This table shows the mapping relation of hop's id and path's id, also shows the flow direction of each hop, (ipsrc and ipdst pairs), idPredecessor shows the last hop's id of current one.

Field	Type	Null	Key	Default	Extra
idHop	int(11)	NO	PRI	NULL	auto_increment
idPath	int(11)	NO	MUL	NULL	
ipSrc	varchar(15)	NO		NULL	
ipDst	varchar(15)	NO		NULL	
idPredecessor	int(11)	YES	MUL	NULL	

Figure 7: the Structure of Hops table

idHop	idPath	ipSrc	ipDst	idPredecessor
1	1	*	*	NULL
2	1	*	*	1
3	1	*	*	2
4	1	*	*	3
5	1	*	*	4
6	1	*	*	5
7	1	*	*	6
8	1	*	100.65.11.161	7
9	1	100.65.11.161	176.32.107.12	8

Figure 8: First 9 lines of of table Hops

Measurements table:

This table shows the mapping relationship between id of measurement and path's id. And it shows the timestamp and rtt (round trip time) = owd_forward_ns + owd_reverse_ns of each measurement.

Field	Type	Null	Key	Default	Extra
idMeasurement	int(11)	NO	PRI	NULL	auto_increment
idPath	int(11)	NO	MUL	NULL	
rtt_ns	bigint(16)	NO		NULL	
owd_forward_ns	bigint(16) unsigned	YES		NULL	
owd_reverse_ns	bigint(16) unsigned	YES		NULL	
timestamp_ns	bigint(16)	YES	UNI	NULL	
pair	bigint(16)	YES		NULL	

Figure 9: the structure of measurement table

idMeasurement	idPath	rtt_ns	owd_forward_ns	owd_reverse_ns	timestamp_ns	pair
1	1	219212143	114552116	104660027	1499538024002969774	NULL
2	1	223202192	118628715	104573477	1499538025003070253	NULL
3	1	219630849	115036556	104594293	1499538029003040135	NULL
4	1	222298329	117612751	104685578	1499538030003193169	NULL
5	1	218995627	114149035	104846592	1499538034002958757	NULL

Figure 10: the first 5 lines of measurement table

Path Table

The path table shows the the mapping relationship between paths and trace direction. Through this table, it could be figured out which path belong to the specified trace direction.

Field	Type	Null	Key	Default	Extra
idPath	int(11)	NO	PRI	NULL	auto_increment
idTraceDirection	int(11)	NO	MUL	NULL	

Figure 11: the structure of table path

idPath	idTraceDirection
1	1
118	1
139	1
212	1
312	1

Figure 12: the first 5 rows of the table Path

TraceDirection Table :

This table contains information about which trace direction is there and what is the source and destination for each one. The columns in this table are idTraceDirections, idSrc and idDst. The first column contains twenty unique ids, there each id represent a source id for a datacenter to destination id of another datacenter. The reason why there are twenty ids is that every source datacenter have four destinations to other datacenters, for example Ireland which has idSrc equals 1 has four destinations which are Tokyo, Oregon, Seoul and N.Virginia. Figure 3 describes the table.

Field	Type	Null	Key	Default	Extra
idTraceDirection	int(11)	NO	PRI	NULL	auto_increment
idSrc	int(11)	NO	MUL	NULL	
idDst	int(11)	NO	MUL	NULL	

Figure 13: **the structure of TraceDirection**

idTraceDirection	idSrc	idDst
1	1	2
2	1	3
3	1	4
4	1	5
5	2	1

Figure 14: **the first 5 lines of TraceDirection**

GeoHops Table

In GeoHops table, it shows the mapping relationship between location's id.

Field	Type	Null	Key	Default	Extra
locId	int(10) unsigned	NO		NULL	
country	char(2)	NO		NULL	
region	char(2)	NO		NULL	
city	varchar(64)	NO		NULL	
postalCode	varchar(5)	NO		NULL	
latitude	float	YES		NULL	
longitude	float	YES		NULL	
startIpNum	int(10) unsigned	NO		NULL	
endIpNum	int(10) unsigned	NO		NULL	

Figure 15: **The structure of GeoHops**

locId	country	region	city	postalCode	latitude	longitude	startIpNum	endIpNum
0					NULL	NULL	0	0
609013	AU	07	Research	3095	-37.7	145.183	16777216	16777471
104084	CN	07	Fuzhou		26.0614	119.306	16777472	16778239
17	AU				-33.494	143.21	16778240	16779263
47667	CN	30	Guangzhou		23.1167	113.25	16779264	16781311

Figure 16: **The first five lines of GeoHops Table**

geoLocation table

Field	Type	Null	Key	Default	Extra
locId	int(10) unsigned	NO	PRI	NULL	
country	char(2)	NO		NULL	
region	char(2)	NO		NULL	
city	varchar(64)	NO		NULL	
postalCode	varchar(5)	NO		NULL	
latitude	float	YES		NULL	
longitude	float	YES		NULL	

Figure 17: THE

locId	country	region	city	postalCode	latitude	longitude
0					NULL	NULL
1	O1				0	0
2	AP				35	105
3	EU				47	8
4	AD				42.5	1.5

2.2 MaxMind

MaxMind provides GeoIp databases and services for customers that want to identify the location for a range of IP addresses [2]. This service is used to extend the provided database with geographical information, that is to get the geographic coordinates for every hop in the database. The choice of using Maxmind as a GeoIP service for extending the database is the fact of its data format which can fit any database and these data can be downloaded as CSV files. There are two files when MaxMind is downloaded. The first one called GeoLiteCity-Blocks.csv. and the second file is called GeoLiteCity-Location.csv. The first file contains IP addresses represented as integers with start and end IP numbers. These can be used later to find if a given IP address can be found in that range. While the second file contains information about where a particular IP address is located

2.3 Visualisation tools

One important aspect of this project is to display geographic information and the result of the algorithm on the world map. We present a little a description for some visualization tools that were considered using for this project. These

2.3.1 Google Maps API

Google provides numerous interfaces which allow embedding of a map into a web page using JavaScript. Google Maps API allows the programmer to display maps on websites and mobile applications. It is designed to work on both mobile devices and web browsers. It is suitable visualization tool since it suits programmers that have beginner or intermediate knowledge of HTML, CSS and JavaScript programming languages [3].Google Maps API is free to use, but the programmer is required to obtain a specific API key. The Key is needed to access the Google Maps Servers and it supports an immense number of users [4]. The API can be used to add markers, polygons and ability to change user's view of a specific map area [5].

2.3.2 Gephi

Gephi is an open-source visualization tool for network visualization and analysis [6]. It helps the user to analyze network graphs in an easy matter, as it offers many visualization options

for any type of network. The real-time visualization is powered by ad-hoc OpenGL engine[7]. Gephi was designed for users that are comfortable with using a visualisation tool, as it has a recognizable feel and looks that makes it easy to pick up and use. Loading a graph data into the system is very easy and it supports different file formats [8]. Graph editing, which includes node insertion and deletion are supported in Gephi.

2.3.3 Mapbox

Mapbox is a freemium web-based visualization tool that enables the users to build maps and powerful applications on a very powerful API[9]. Mapbox graphical interface is easy to navigate for users with some background in photoshop and illustrator may find it very easy to use [10]. MapBox is simple to use and contains lots of styling options for the maps. these styling options comes in form of customizations for how to display streets, buildings, and terrain with different colors and the ability to add labeled markers with different sizes and styles [11].

2.4 Round Trip Time (RTT)

In computer networks, RTT is the elapsed time between the time a packet is sent by the source to the time the corresponding ack is received by the source[12]. It includes propagation, queuing, processing and other delays at the routers and the end node [12]. RTT highly affects the utilization of network paths. Since high RTT leads to high queuing delay, which results in high dropping possibility and low utilization [13]. In a network, especially a WAN or the Internet, RTT is one of the main factors that influence the latency [14].

2.5 TraceRoute

Traceroute was originally developed with the intention of providing a quick and effective debugging tool, which could be used to help determine which device or segment of the network may be causing network problems. Traceroute will display IP addresses of routers that the IP networks pass by[15].

Traceroute has three main features:

1. It is Cross-platform.The Traceroute tools can work on various operating system platforms, including mainstream MAC OS, Windows, Linux, Android, IOS, etc.
2. Easy to use. We just need to enter IP or domain name.
3. It has comprehensive information. Traceroute can display the number of jumps, packet loss, delay, etc

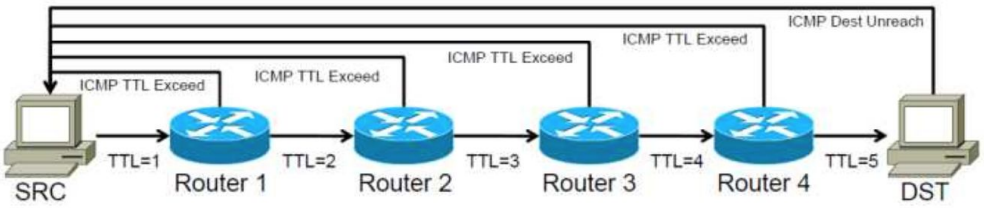


Figure 18: Traceroute implementation principle[15].

The implementation principles of traceroute.

1. Send a probe from SRC to DST and set TTL to 1;
2. The TTL will be reduced one by each time.
3. When TTL becomes 0, the packet is discarded, and the router sends an ICMP TTL package to SRC.
4. When SRC receives the ICMP package, then this frame of information is displayed;
5. Repeat 1 ~ 5, and TTL plus 1 each time;
6. Until the DST received packet, and returns the ICMP Dest Unreachable package;
7. When the SRC received ICMP Dest Unreachable package, we will stop traceroute.

DNS reverse resolution in Traceroute

We can obtain the following information form DNS reverse resolution in Traceroute:

1. The Location of the router
2. Interface type and bandwidth
3. Router types and roles
4. The boundary and relation of network autonomous system

2.6 Dijkstra's algorithm

Dijkstra's algorithm, is an algorithm about graph search with non-negative edge path costs, producing a shortest path tree.For a given source node in the graph, the algorithm could find the path which has lowest cost between one node to the rest of other nodes. It can also be applied in finding costs of shortest paths from a single node to a single destination node by stopping the algorithm once, the shortest path to the destination node has already been determined [16].This algorithm could be often applied in routing and as a subroutine in other graph algorithm. It is usually the working principle behind link-state routing protocols, OSPF and IS-IS being the most common ones [17].

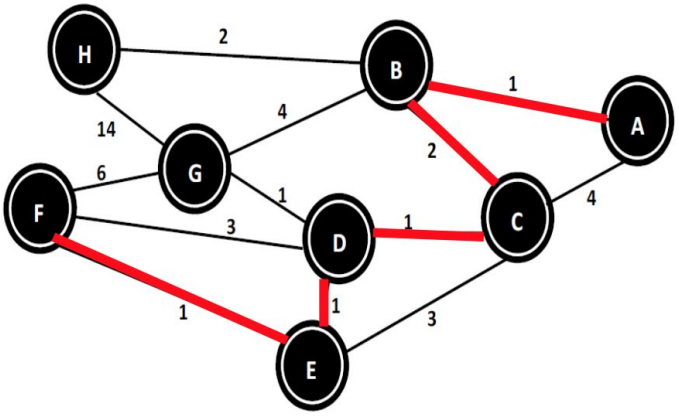


Figure 19: Dijkstra's algorithm network of 8 nodes (source: A destination:F)

	(d, n)	(d, n)	(d, n)	(d, n)	(d, n)	(d, n)	(d, n)
cluster	B	C	D	E	F	G	H
A	1, A	4, A	∞	∞	∞	∞	∞
AB		3, B	∞	∞	∞	5, B	3, B
ABC			4, C	6, C	∞	5, B	3, B
ABCH			4, C	6, C	∞	5, B	
ABCHD				5, D	7, D	5, B	
ABCHDE					6, E	5, B	
ABCHDEG					6, E		
ABCHDEGF							

Figure 20: the routing table for node A

Running time of Dijkstra's algorithm

- $O(V^2)$ − In order to obtain the routing table, we need O(V) rounds iterations (until all the vertices are included in the cluster). In each round, we will update the value for O(V) vertices and select the closest vertex, so the running time in each round is O(V). So, the total running time is $O(V^2)$.

Disadvantages of Dijkstra's algorithm

- If the scale of the network is too large, then it will cost a long time to obtain the result.

- For some time-sensitive app or real-time services, we need to reduce the running time.

2.7 Latency

Network latency is defined as the time delay observed in data transmits from one network node to another, where the source and destination are used [18]. There are lots of factors that contribute to network latency, including transmission, propagation, routers and computer hardware delays.

In network latency, transmission refers to the medium used to transmit the data [18]. For example, this may be a phone line, fiber optic line or wireless connection,etc.

Propagation delay is one of the harder things to control in network latency [19].

This is in proportion to the physical distance between the origin and destination. Naturally, the greater the distance, the more delayed the transmission will be. However, this does not usually cause a significant delay [19]. The latency from routers and computer hardware, may be able to be changed. In this case, upgrading corresponding middleware can help processing information faster, thus speeding up the process. However, this may involve a substantial investment. This is supposed to be reconsidered, depending on how much and how often data is transferred [19].

2.8 Background relevance to the project

Dijkstra's algorithm, RTT, latency, traceroute, and visualization tools that have been described, discussed and defined in this section are crucial to this project. These elements are of a recurrent nature throughout the course of the project. It is therefore of the utmost importance that they are understood in their entirety before heading to the next section. The following sections will, therefore, assume that these elements have been fully understood in advance.

3. Research methods

This section describes the research methods that have been used in this project. This section also provides an in-depth explanation of the different research phases that have been used. This chapter is essential as it gives the reader a deeper understanding of how the project has been conducted.

3.1 Method

Due to the nature of the project, the quantitative research method would be best suited but with some elements of the qualitative research. The quantitative research will provide the necessary tools to analyze data to make plots for different parts of the project and to also help with checking which visualization tool would perform better from one another. Although quantitative research is the method primarily used in the project, a research for finding the correct and the most suited visualization tool for the project has to be done. This required a qualitative research method in order to interpret documents to find the correct visualization tool that meets all the requirements and functionalities necessary for project completion.

The first research method is quantitative research, which is an empirical research study of observable phenomena via statistical and mathematical methods. The method is used to analyze data or to quantify the problem by generating numerical data that can be converted into useful statistics [20][21].
The second research method is qualitative research, which is an interpreting research that is applied in various academic disciplines, such as social science or natural science, but also within market analysis and other similar contexts. The strength of qualitative research is its ability to provide a textual description of how people experience a particular research question [22].

3.2 Research phases

This project is divided into three different phases; literature study, coding and evaluation. The literature study was conducted to understand more about the project and what is needed to accomplish. The coding part is the most important since we need to fetch data from the database and analyze these data, create topologies and implementing both the backend and frontend. The evaluation part included evaluating the EICS

3.2.1 Literature study

A literature study has been conducted to understand more about the subject at hand. We started by reading relevant materials and resources about reducing WAN latency. We also did a literature study for finding the correct visualization tool to be used to display information from the database onto the world map. A literature study was also conducted to find suitable geographic IP services. Choosing MaxMind as a GeoIP service because there are lots of documentation on how to use, implement and extend it to an existing database. some parts of this project required to know where a datacenter and hops are physically

located. The easiest way to do that is by looking up their IP addresses in a special database called MaxMind. In the result section, we will dive in details on how to extend the existing database with MaxMind geolocation.
Google Maps API was chosen after a literature study of the three most used visualisation tools that can be suitable for this project. A comparison between these visualisation tools has been done to motivate our decision.

3.2.2 Coding

Coding different parts of the project required knowledge in different programming languages. We have used SQL language to create and filter the database tables. JavaScript was mostly used for developing the interactive visualization system (IVS) for this project. Java programming language was used to implement Dijkstra's algorithm, the server that acts as an interface between the IVS, and the expert infrastructure change system (EICS). Other programming languages have been considered but some members of this project are not that knowledgeable in them.

3.2.3 Evaluation

In the Literature study, we focused primarily on finding the correct visualization tool and geolocation service in order to provide sufficient knowledge in the area that we found to be the most important for completing the project. We started by studying Google Maps API, Gephi, and Mapbox visualization tools since they are the most suitable options for this project. The reason why these visualization tools were chosen because they are frequently used and had lots of online documentations. These tools were also easy to implement and develop since there are lots of implementation that make use of these tools. Furthermore, these tools are recognized by many experts in the field of visualization, thus frequently used for visualization.
Regarding the geolocation services, We found that most of these tools offer the same functionality and in majority quite similar to each other. Finding one was rather easier than anticipated. But these services are not without problems. Problems were present when trying to extend the database with geographic information. The data that these services provide were not that precise for the free version. To get more precise results required paying a monthly or annual fee.
Evaluating the algorithm performance required running the algorithm with different topologies of different sizes, specifying a limit for the total distance of the network links, and how many network hops (nodes) is allowed to be introduced to the topology. These are all important factors when evaluating our algorithm since the results may differ when one of these parameters changes. We will also evaluate the complexity of the algorithm and ways of reducing complexity.

3.3 Evaluation Criteria

To evaluate and compare the various visualization tools an evaluation criterion was presented. An evaluation criterion is important for highlighting the different aspects of each visualization tool while providing a structured way of assessing strengths and weaknesses. Criteria that are being used in the evaluation are partially based on the literature study that was conducted when trying to find a suitable visualization tool. There are also criteria that were mentioned by other researchers/developers that we thought they should be included. There were some evaluation criteria that were considered less important for the advantages and disadvantages of the visualization tools.

The first criteria deal with the ability to displaying graph(s) on the world map. All visualization tools are capable of achieving this criterion but they have different approaches to display these graphs. The second criteria deal with adding vertices and edges of different size, shapes, and colors to the graph and display it on the world map. All three visualization tools are capable of modifying and customizing the graph with ease, but some are better than the others. The third criteria was important in the beginning of the project since the database has been an important aspect of this project, but due to a problem that occurred during database extension, it became less important for completing this project. However, all visualization tools offer database functionalities.
The fourth, fifth, and sixth criteria deal with user interactivity, usability, and documentation. We found that in these three criteria there are some differences that led us to choose one out of these tools.

Obviously, there were more criteria that can be used to evaluate these tools, but they were not included to prevent the evaluation of becoming too extensive. The mentioned criteria formed the basis of what the project considers to be a versatile tool for visualization graph onto the world map.

4. Results

To achieve the desired results for this project needed some important parts of project to be researched, implemented and developed. We started the project by analyzing, understanding, and interpreting the dataset generated by the traceroute. This will give enough knowledge to know how to interact with MySQL database and interpret network path information, which in it turn will help with analyzing the data of how individual measurements can be formed into overall network topology.
While we were analyzing the database, we were also designing, developing, and setting up the visualisation system that contains some GUI elements that the user can interact with and will display different network topologies. The final part of the project deals with development of an expert infrastructure change system that can analyse the data, suggest routing infrastructure changes and display the resulting infrastructure changes through the visualisation system.

4.1 Network path data analysis

Data analysis of network paths requires that the MySql database has been set up on the virtual machine and the dataset that has been provided by the traceroute has been imported successfully on the virtual machine. This will help with completing subsections 1.1 through 1.4 in the project

4.1.1 Demonstrate that the number of network paths between any two datacenters is finite.

To complete this part will require to thoroughly analyze every table in the database, what is the relationship between every table, and what does every table contain. Figure 1 displays the relationship between every table in the database and their content. Afterwards, filtering these tables using SQL language is necessary to analyze tables that contains information about paths, trace direction and timestamps. There every trace direction has multiple paths and every path has timestamps. Afterwards, a plot has been made which contains information about how frequent new paths are being observed with respect to their times.
The figure below shows first 9 lines of table Measurement

idMeasurement	idPath	rtt_ns	owd_forward_ns	owd_reverse_ns	timestamp_ns	pair
1	1	219212143	114552116	104660027	1499538024002969774	NULL
2	1	223202192	118628715	104573477	1499538025003070253	NULL
3	1	219630849	115036556	104594293	1499538029003040135	NULL
4	1	222298329	117612751	104685578	1499538030003193169	NULL
5	1	218995627	114149035	104846592	1499538034002958757	NULL
6	1	221754911	116967459	104787452	1499538035003061559	NULL
7	1	218851308	114067113	104784195	1499538039002956974	NULL
8	1	221653277	116824716	104828561	1499538040003058630	NULL
9	1	218853533	113998327	104855206	1499538044002947215	NULL

Figure 21: The content of database table Measurements.

1. Filter all the idPath(s) belong to each pair of datacentres, for instance idNodes 1 and 3 (Ireland and Oregon), all idPaths are in idTraceDirection equals 2 and 9. The

timestamp_ns has been divided by 1000000000 for better calculation.The table below shows the path id and the start time (minimum timestamp of each path) in data centres pairs from Ireland to Oregon.

2. Then select the minimum of **startime** of the new table above to find the very beginning of when new paths appear.
3. For each hour increase, the **startime** (timestamp_ns div 1000000000) increase 3600.In that case we make a loop to measure how many paths appear per hour.
4. The appendix show the data about how many new paths occurs per hour with respect to each data centres pairs. Figure 8 and 9 shows the obtained results.

idPath	startime
1	1499538024
28	1499537974
29	1499537994
35	1499538179
42	1499538374
64	1499539114
67	1499539119
71	1499539159
118	1499540520
128	1499541234
129	1499541259
139	1499541755
150	1499541949
152	1499541980
158	1499542164

Figure 22: the filtered result.

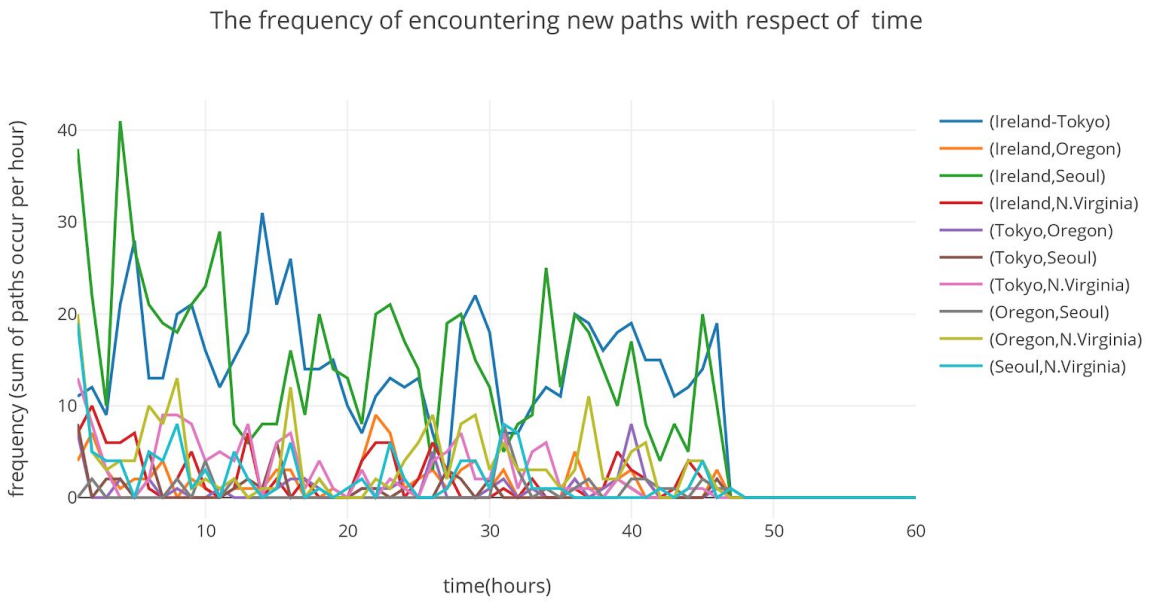


Figure 23: The frequency of encountering new paths with respect to time

4.1.2 Analyze persistence of network paths.

This subsection is about producing a CDF plot for a pair of datacenters that shows the average persistence of network paths.
The results were achieved by doing the following steps :
Take id TraceDirection = 12 (N.Virginia to Oregon) as an example.

1. Filter all Paths exists in table Measurements from N.Virginia to Oregon and create a table for it. Afterwards, select the first nine rows as an example and alter an AUTO_INCREMENT id to the table for better calculation.

id	idMeasurement	idPath	time
1	218	7	1499538028
2	219	7	1499538032
3	220	7	1499538033
4	221	7	1499538037
5	222	7	1499538038
6	223	7	1499538042
7	224	7	1499538043
8	225	7	1499538047
9	226	7	1499538048
10	227	7	1499538052
11	228	7	1499538053
12	229	7	1499538057
13	230	7	1499538058
14	231	7	1499538062
15	232	7	1499538063
16	233	7	1499538067
17	234	7	1499538068
18	235	7	1499538072
19	236	7	1499538073
20	237	7	1499538077

Figure 24: The table filtered by all paths flow from N.Virginia to Oregon.

2. Calculate the sum of all paths exists from from N.Virginia to Oregon as Sum1.
3. Add another time column as **to_time** which is one row downside compared to **from_time** (**from_ time** column is exactly the time column of the last table in step 1). **From _time** minus **to_time** as duration. Add src column (as same as **idPath**), add **dst** column =(current id +1).**idPath**. When **src=dst**, (means the calculation in the same path so far) sum all duration as each path persistence of each idPath.

id	idPath	src	dst	from_time	to_time	duration
1	19	19	19	1499538031	1499538036	5
2	19	19	19	1499538036	1499538037	1
3	19	19	19	1499538037	1499538041	4
4	19	19	19	1499538041	1499538042	1
5	19	19	19	1499538042	1499538047	5
6	19	19	19	1499538047	1499538051	4
7	19	19	19	1499538051	1499538056	5
8	19	19	19	1499538056	1499538057	1
9	19	19	19	1499538057	1499538061	4
10	19	19	19	1499538061	1499538062	1

Figure 25: The first ten rows of the table shows every persistence of path.

4. Sort out all idpaths belong to different time_frame(MOD(sum of duration of each path, 3600 Seconds.))

idPath	time_frame
19	21
22	21
30	19
32	18
65	24
66	24
69	24
78	17
79	17
103	14
115	14
122	13
131	16
132	3
153	0
195	22
222	20
264	0
301	21
339	11

Figure 26: The first 20 idPaths results shows the idPath and the sum of persistence belongs to each path.

5. There's some idPaths only has one idMeasurement, which means only has one single timestamp. Timestamps in database is accurate to nanoseconds, which assumes that the persistence is 0 here. Insert into the pdf and cdf values of paths which has persistence of 0 seconds.Assume the sum of all paths in last table as Sum2 ,Sum1- Sum2 = sum(all paths have 0 time persistence here). (Count(idPath) of each time_frame)/ Sum1 is pdf of average persistence of each paths. $cdf = \int pdf \ dt$. The figure 10 shows the first 20 rows of the pdf and cdf of the paths.

persistence	pdf	cdf
0	0.0385	0.0385
1	0.0256	0.0641
2	0.2179	0.2820
3	0.0833	0.3653
4	0.0513	0.4166
5	0.0833	0.4999
6	0.0385	0.5384
7	0.0256	0.5640
8	0.0321	0.5961
9	0.0256	0.6217
10	0.0128	0.6345
11	0.0192	0.6537
13	0.0128	0.6665
14	0.0064	0.6729
16	0.0128	0.6857
19	0.0064	0.6921
20	0.0064	0.6985
24	0.0064	0.7049
26	0.0064	0.7113
27	0.0064	0.7177

Figure 27: The first 20 rows of pdf and cdf.

6. We did the same things above for each trace and plot the cdf of average persistence of paths in each pair data centres. The appendix gives two csv files shows the the data collected for each path's pdf and cdf of average persistence.

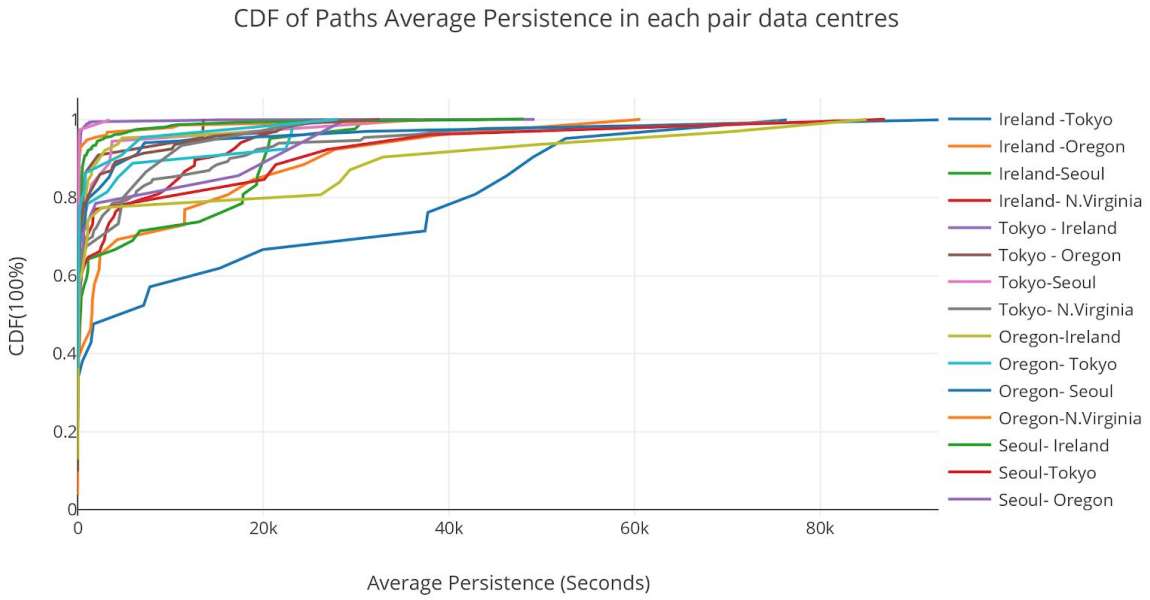


Figure 28: CDF plot for average paths persistence in each pair data centers.

4.1.3 Extend measurement database with geographical information.

Extending the database requires finding an appropriate IP geolocation technique to extend our database with geographic information. Afterwards, we look into a way to translate every IP address in the database into information containing geo location and store that information in a new table. This will help us later on when designing and developing an interactive visualizations tool. We choose to work with MaxMind[23], since when using CSV files provided by MaxMind, we noticed that the files are just data, there are no database schemas to fit the data on the database. This makes them suitable to be included in any database system [24]. We created two tables in our database, one is called geoLocation that contains information regarding the cities, latitude and longitude, and the other table GeoIP contains ip addresses which corresponds to their locations.

To find the locations of hops, we have looked for source IP addresses in table **Hops** that matches **start and end ip number** in table **GeoIp** and then get the geographic information from **geoLocation** table.

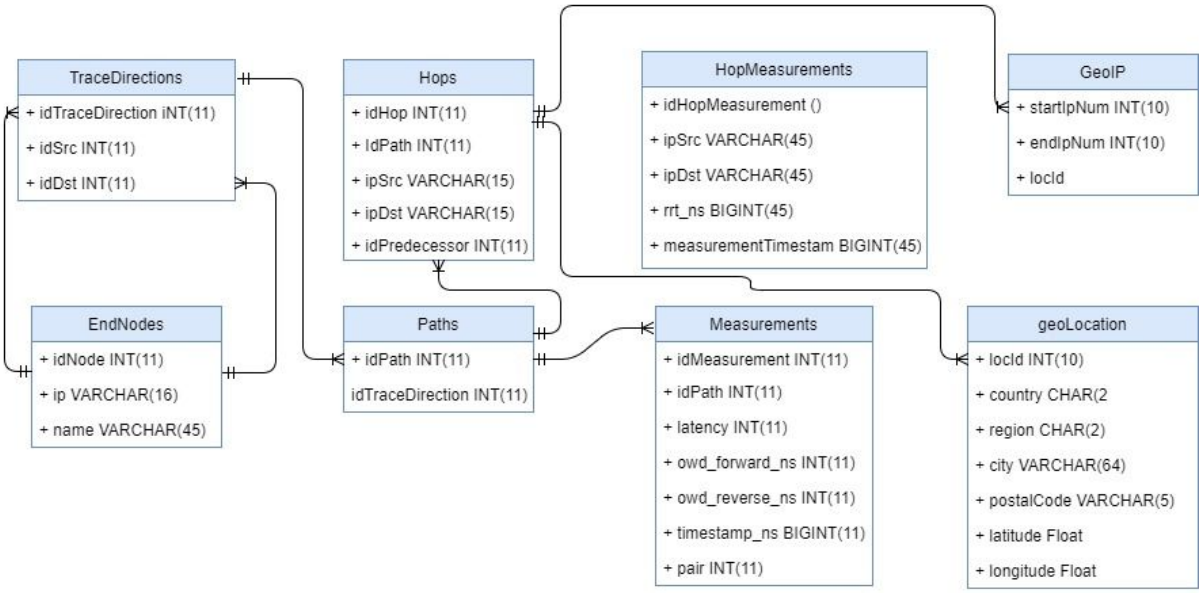


Figure 29: Extending the database with geographic location.

Creating the tables:

The following commands have been used to create the database tables in MySql database. These tables will be filled in with data from MaxMind dataset found in the following files : GeoLiteCity-Blocks.csv. and GeoLiteCity-Location.csv.

```
mysql> CREATE TABLE GeoIp (
    -> startIpNum int(10) unsigned NOT NULL,
    -> endIpNum int(10) unsigned NOT NULL,
    -> locId int(10) unsigned NOT NULL,
    -> PRIMARY KEY (startIpNum, endIpNum),
    -> );
```

```
mysql> describe GeoIp;
+------------+------------------+------+-----+---------+-------+
| Field      | Type             | Null | Key | Default | Extra |
+------------+------------------+------+-----+---------+-------+
| startIpNum | int(10) unsigned | NO   | PRI | NULL    |       |
| endIpNum   | int(10) unsigned | NO   | PRI | NULL    |       |
| locId      | int(10) unsigned | NO   |     | NULL    |       |
+------------+------------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Figure 16: Creating GeoIp table that will contain information from GeoLiteCity-Blocks.csv.

Figure 30:

```
mysql> CREATE TABLE geoLocation (
    -> locId int(10) unsigned NOT NULL,
    -> country char(2) NOT NULL,
    -> region char(2) NOT NULL,
    -> city varchar(64) NOT NULL,
    -> postalCode varchar(5) NOT NULL,
    -> latitude float,
    -> logitude float,
    -> PRIMARY KEY (locId));


mysql> describe geoLocation;
+------------+------------------+------+-----+---------+-------+
| Field      | Type             | Null | Key | Default | Extra |
+------------+------------------+------+-----+---------+-------+
| locId      | int(10) unsigned | NO   | PRI | NULL    |       |
| country    | char(2)          | NO   |     | NULL    |       |
| region     | char(2)          | NO   |     | NULL    |       |
| city       | varchar(64)      | NO   |     | NULL    |       |
| postalCode | varchar(5)       | NO   |     | NULL    |       |
| latitude   | float            | YES  |     | NULL    |       |
| longitude  | float            | YES  |     | NULL    |       |
+------------+------------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Figure 31: Creating the table that will contain information from GeoLiteCity-Locations.csv.

The IP addresses found in GeoLiteCity-Blocks database are represented as integers. This will allow to easily compare these addresses as number to determine if a given IP address falls into the given range. We use the function **INET_ATON** to interpret the ip addresses into into integer representation [25]. Figure 7 shows in details IP address translation to integer representation .

```
mysql> SELECT * FROM GeoIp JOIN geoLocation ON geoLocation.locId = GeoIp.locId
WHERE startIpNum <= INET_ATON('52.95.30.131') ORDER BY startIpNum DESC LIMIT 1;
+------------+------------+--------+--------+---------+--------+-------+------------+----------+-----------+
| startIpNum | endIpNum   | locId  | locId  | country | region | city  | postalCode | latitude |
longitude |
+------------+------------+--------+--------+---------+--------+-------+------------+----------+-----------+
|  878648832 | 878649343 | 875630 | 875630 | JP      | 13     | Tokyo | 102-0      |   35.685 |
139.751 |
+------------+------------+--------+--------+---------+--------+-------+------------+----------+-----------+
```

Figure 32: Translate and get the location of a given IP address.

Figure 33: Confirming location of an IP address [26].

We have made a test for a given source ip address found in our database to see if it can be found in Maxmind database. The above figure 7 shows the sql query that have been used to interpret the source ip address “52.95.30.131” and the results shows that it is located in Tokyo Japan. The second figure 8 is for confirmation that the Ip address is indeed located in Tokyo Japan.

Next step will be to link our Hops table to the GeoIp table to be able to easily interpret any source and destination ip address with some geographical information.

```
mysql> SELECT *  FROM Hops, GeoIp JOIN geoLocation On geoLocation.locId = GeoIp.locId WHERE
startIpNum <= INET_ATON(ipSrc) AND idHop = 13572 ORDER BY startIpNum DESC LIMIT 1;
+-------+--------+-------------+------------+--------------+------------+------------+-------+-------+---------+--------+--------+
| idHop | idPath | ipSrc       | ipDst      | idPredecessor | startIpNum | endIpNum   | locId | locId | country |
region | city   | postalCode | latitude | longitude |
+-------+--------+-------------+------------+--------------+------------+------------+-------+-------+---------+--------+--------+
| 13572 |    573 | 176.32.107.6 | 52.93.36.86 |        13571 | 2954913792 | 2954915839 | 20632 | 20632 | IE
| 07    | Dublin |              |  53.3389 |   -6.2595 |
+-------+--------+-------------+------------+--------------+------------+------------+-------+-------+---------+--------+--------+
```

Figure 34: Testing if a given idHop source IP can we get the location of that Ip address

The above figure 19 is for testing if a given idHop can we find the location of the ip source address.

Unfortunately this implementation did not work as we thought it would. The result the we got was not that useful for this project since it showed only ten geographical hops where in fact there are many more. The problem is with the provided dataset, because when translating the ip addresses to geographic information they were mapped to hidden locations. Due to this problem, we came up with new implementation to get the desired results for this project. The new implementation will be described in the result section of the report.

4.1.4 Correlate network latency with geographical distance.

We have made some mathematical correlation and we anticipate that there is a direct relationship between network latency and geographical distance. It implies that in most cases when geographical distance increases the network latency will also increase. However in some unusual cases this equation can be more complex and sophisticated.
When mentioning network distance among datacenters, if a plot (diagram) has two axes (aspects or dimensions) called X and Y then we can have a pair of (X,Y) where the X-axis is the geographical distance and the Y-axis is the corresponding network latency related to that geographical distance, or vice versa. In this case (vice versa) we can change the name of the X-axis to “t” to represent a time measurement about network latency.

1 The number of intermittent IP address are seen between any pair of datacenters

1) Create a temporary table that contains the idPath, idHop, idTraceDirection

```
mysql> create temporary table kat
    -> select t1.idPath, t1.idHop, t2.idTraceDirection from Hops t1
    -> left join Paths t2 on t2.idPath = t1.idPath
    -> union select t2.idPath, t1.idHop, t2.idTraceDirection from Hops t1
    -> right join Paths t2 on t2.idPath = t1.idPath;
```

2) Create temporary tables that show the number of hops of every path in a pair of datacenters.

```
mysql> create temporary table cou2 select idPath, count(*) as numero from
kat where idTraceDirection = '2' group by idPath;
```

3) Count the number of paths with the same number of hops (network distance)

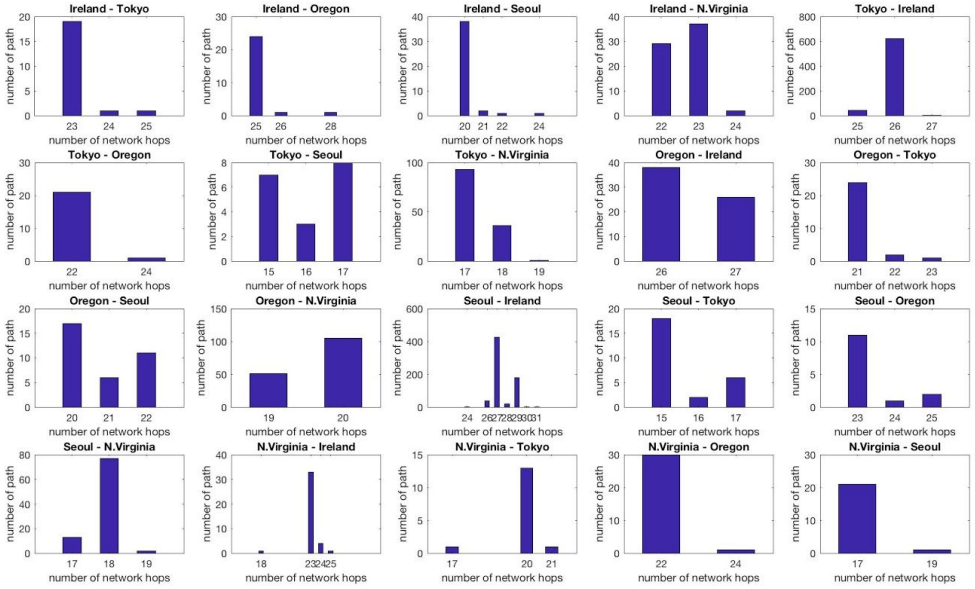
```
mysql> select numero, count(*) from cou2 group by numero;
+--------+----------+
| numreo | count(*) |
+--------+----------+
| 25     |    24    |
| 26     |     1    |
| 28     |     1    |
+------------+------+
```

4) Use these numbers to produce a plot by using Matlab

```
figure

subplot(4,5,1)
x = [23 24 25];
y = [19 1 1];
bar(x,y,0.5)
xlabel('number of network hops')
ylabel('number of path')
title ('Ireland - Tokyo')
```

5) Result



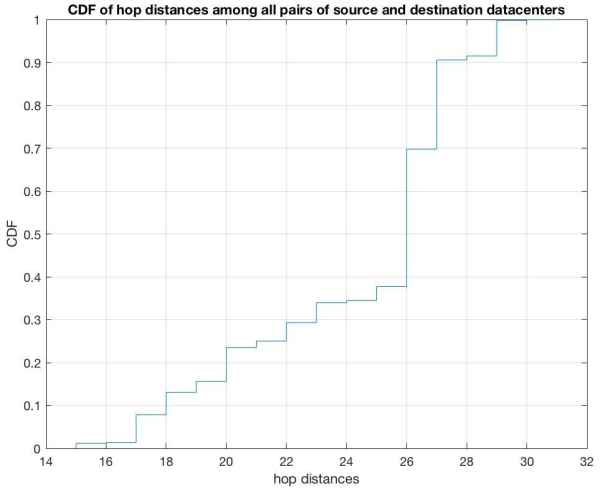
This result shows that the network distance among different pair of datacenters are diverse.

2 CDF shows hop distances among all pairs of datacenters

Calculate the total number of certain network distance among all pairs of datacenters manually, then plot a CDF by using Matlab.

```
a = [15 25
    16 5
    17 142
    18 114
    19 55
    20 173
    21 33
    22 94
    23 101
    24 12
    25 71
    26 700
    27 454
    28 21
    29 181
    30 3
    31 2];
b=[];
for i=1:length(a(:,1))
    b=[b;a(i,1).*ones(a(i,2),1)];
end
cdfplot(b)
xlabel('hop distances')
ylabel('CDF')
title('CDF of hop distances among all pairs of source and destination datacenters')
```

result



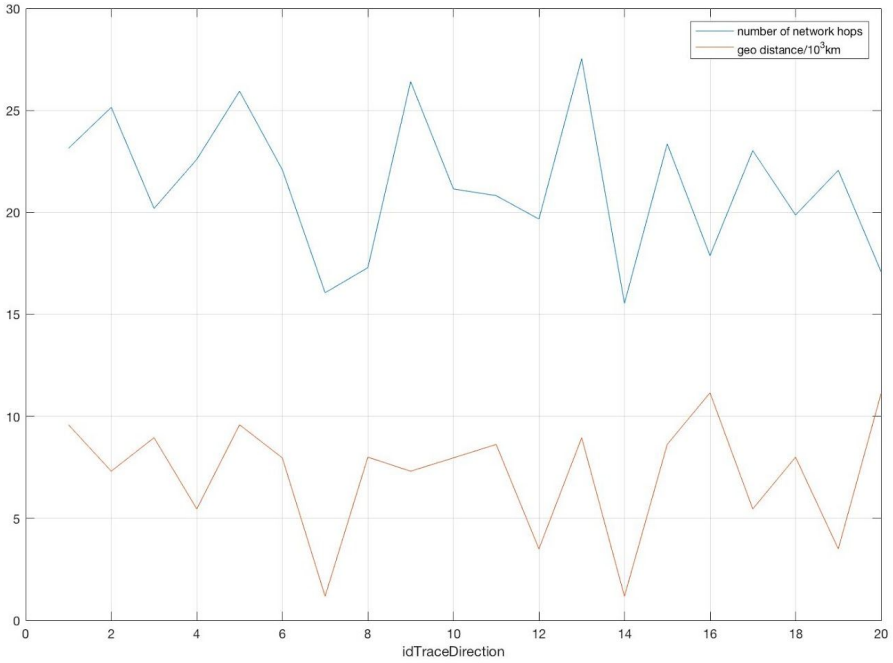
This result shows that among all datacenters, the longest paths have 29 hops while the shortest paths have 15 hops.

3 Correlation between number of network hops and geographic distance

```
y1 = [23.14 25.15 20.19 22.60 25.94 22.09 16.06 17.29 26.41 21.15 20.82 19
y2 = [9.587 7.312 8.954 5.460 9.587 7.965 1.178 8 7.312 7.965 8.624 3.502
plot(y1)
hold on
plot(y2)
hold off
legend('number of network hops','geo distance/10^3km')
xlabel('idTraceDirection')
grid on
```

y1 is the average hops number of each pair of datacenters, while y2 is the geographic distance of each pair of datacenters. By entering the IP of datacenters, the latitude and

longitude would be given by the database. Then the distance is measured by using google map, and the distance is the direct distance on the surface of the Earth.
result



4 Correlation between network latency and geographic distance.

1) Create a temporary table that contains the idPath, rtt, and idTraceDirection.

```
mysql> create temporary table fyra
    -> select t1.idPath, t1.rtt_ns, t2.idTraceDirection
    -> from Measurements t1
    -> left join Paths t2 on t2.idPath = t1.idPath
    ->union
    ->select t2.idPath, t1.rtt_ns, t2.idTraceDirection
    ->from Measurements t1
    ->right join Paths t2 on t2.idPath = t1.idPath;
```

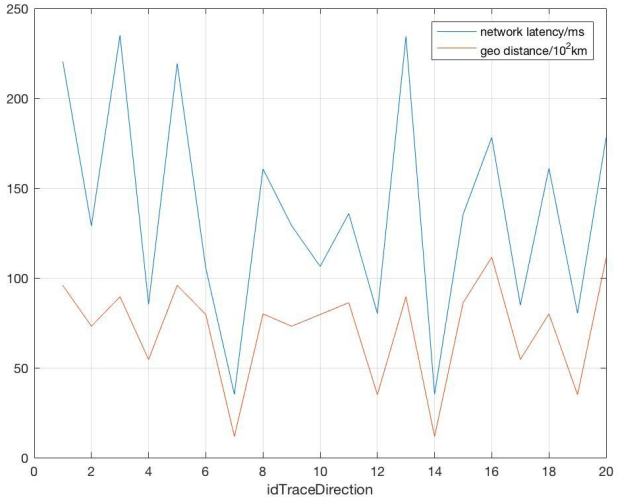
2) Calculate the average latency(rtt) of every pair of datacenters.

```
mysql> select idTraceDirection, avg(rtt_ns)/1000000 as lat_ms
    -> from fora group by idTraceDirection;
+------------------+--------------+
| idTraceDirectio  |     lat_ms   |
+------------------+--------------+
|                1 | 220.45500034 |
|                2 | 128.92266075 |
|                3 | 234.97095400 |
|                4 |  85.31575045 |
|                5 | 219.25081181 |
|                6 | 105.46276225 |
|                7 | 35.20412035  |
|                8 | 160.63133675 |
|                9 | 129.09026056 |
|               10 | 106.38153274 |
|               11 | 135.80414138 |
|               12 |  80.21536216 |
|               13 | 234.33323873 |
|               14 |  35.27509535 |
|               15 | 135.46240362 |
|               16 | 178.18756049 |
|               17 |  84.83720975 |
|               18 | 160.98660787 |
|               19 |  80.30867161 |
|               20 | 178.15499137 |
+------------------+--------------+
```

3) Produce a plot. Y1 is the latency while ye is the geographic distance of each pair of datacenters.

```
y1 = [220.45500034 128.92266075 234.97095400 85.31575045 21
y2 = [95.87 73.12 89.54 54.60 95.87 79.65 11.78 80 73.12 79
plot(y1)
hold on
plot(y2)
hold off
legend('network latency/ms','geo distance/10^2km')
xlabel('idTraceDirection')
grid on
```

4) Result



This result shows that the network latency and geographic distance are related.
According to the result of 3 and 4, it is clear that geographic distance is not related to the number of network hops but related to the network latency. Therefore, it can be assumed that by adding more hops into two known hops, the latency can be decreased.

4.2 Design and Develop Interactive Visualisation System (IVS)

The sections deal with designing, developing, and setting up the visualisation system. To complete the desired results, we will need to do some literature study to understand more about visualisation tools, their features, required plugins, and maybe also IP geolocation techniques.

4.2.1 Research and choose a visualisation tool for the project.

After performing the literature study to search for visualization tool that meet the IVS requirements. The results can be viewed in the tables below, where we looked at the advantages and disadvantages of using each visualization tool.
There are 3 different tools that we focus on: Gephi, Google Maps API and MapBox:

Table 1: Gephi advantages and disadvantages [27]

Using this tool, is it possible to...	Yes No	**How this may be fulfilled?**
...display a graph on a world map?		Gephi offers plugins that allow the user to add a world map to the interface using a GeoLayout plugin [28]. This plugin will display the graph based on the geocoded latitudes and longitudes.
...add vertices and edges of different size?		Gephi allows graph customization with different colors and shapes to represent a network[29].

...retrieve information from the database		“Gephi can read wide variations of graph formats and supports the reading of CSV files and imports of relational databases”[29]
…change dynamically the rendering according to user input (being able to select from and to which network hops the network paths are being displayed)		Gephi allows the user to filter and select nodes and/or edges based on the network. Gephi’s interactive user interface enables the user to filter the network in real-time[29]
… have an easy-to-use documentation		Gephi does not provide an easy-to-use documentation
… user interaction		Gephi demands to use a java plugin to add user interaction to the map

Table 2: Google Maps API advantages and disadvantages [30]

Using this tool, is it possible to...	Yes No	**How this may be fulfilled?**
...display a graph on a world map?		Lots of documentation on how the display the graph with nodes and draw links between these nodes on the graph [31].
...add vertices and edges of different size?		Using a geolocation API to add vertices into the map [32]. Customize the size of the vertices and edges [33].
...retrieve information from the database		Easy to display information from a database onto the map using Google Maps JavaScript API [34].
…change dynamically the rendering according to user input (being able to select from and to which network hops the network paths are being displayed)		Using javaScript we can easily trigger functions that enables dynamic rendering [34.]
… provide full features		some features are missing
… provide systematic IDE		Does not afford systematic IDE

Table 3: MapBox advantages and disadvantages [35]

Using this tool, is it possible to...	Yes No	**How this may be fulfilled?**
...display a graph on a world map?		MapBox contains lots of services that enables the user to add maps and customize these maps using their API [35].
...add vertices and edges of different size?		The Mapbox API has lots of customizations that allows adding vertices with different shapes and sized and edges linking these

		edges to each others [35]
...retrieve information from the database		MapBox support retrieving information from the database using the Datasets API. This API involves the interaction between two types of resources; datasets (contains information about the dataset) and features [35].
…change dynamically the rendering according to user input (being able to select from and to which network hops the network paths are being displayed)		Can be easily done using their provided geographic coordinates in their API
… provide documentation		Lots of documentation for every necessary part that the user may want with graphs [35].
… provide GUI		Does not provide an integrated GUI
… retrieving data		Implies using a back-end in addition to front-end to retrieve the data

Google Maps API and Gephi have been selected to be tested for basic prototyping, in order to figure out which one would be the more efficient to perform this part of the project. Gephi has been put aside, because coding user interactivity for Gephi would imply to use it as a module in a Java project, and Java is not a language any of us is very at ease with.
We chose Google Maps as a tool to build the IVS, for its rich mapping features as for the easiness to build upon such an API, since it's a web-based API.

Evaluation on Google map API:

Advantages:

- Google Maps API can meet all requirements and functionality necessary for the completion of our project with its rich mapping features. (Main reason)
- It's free and suitable for lots of Web browsers.
- Data from Google Maps API can give us accurate real-time information for mapping, plotting and locating.
- Compared with other visualisation tools, Google Maps API is easier to develop with, because of its detailed Doc instructions, Github resources, debugging method especially under stackoverflow "Google Map" label.
- It's a web-based API. Some of our group members have experience on web development, which makes easy to build the IVS. For phase 2, we already used Google Maps Javascript API and Google Maps Elevation API, coding with Javascript, PHP, etc.

Disadvantages:

- Compared with some other visualisation tools, Google Maps API doesn't afford systematic IDE, which means we cannot develop with GUI. Thus, all the deployment

with Google maps can only be realised by programming or CLI interaction, such as cURL request.
- It still lacks some specific features or methods. For example, given the path, we cannot sample points on it with Google Maps Javascript API, that's why we apply Google Maps Elevation API instead.

4.2.2 Design and develop static IVS prototype.

Following subsection 2.1, we started by designing and implementing a prototype for the IVS. The visualisation system should read network path information from the database, construct topology graph and display it on the world map. Therefore, the prototype should be able to produce a single graph (image) based on the static configuration.
We were given some requirements that needs to implemented. These requirements are as follows:

- All measurement endpoints (i.e., datacenters) should be placed correctly at their geographic locations.
- All identified IPs should be placed at the correct geographic locations.
- IPs that are common for multiple trace directions must be highlighted as such the width of the edges should be proportional to the popularity of the given segment.

4.2.2.1 Simple prototype of IVS

For Google Maps API, we start to make a working prototype as per requirements of visualisation tools by using Google Maps Javascript API. We first try to add nodes and edges onto the world map, and then we search for the datacenter locations in a database. The search is done by using the city name of where the nodes are located and map it with its geolocation using Google Maps. We can choose to display the nodes as regular or as a custom icons and show an information windows that contains a nodes location when we click on it. We also added a few paths onto the world map using the *Polyline* object using Google Maps API [36].

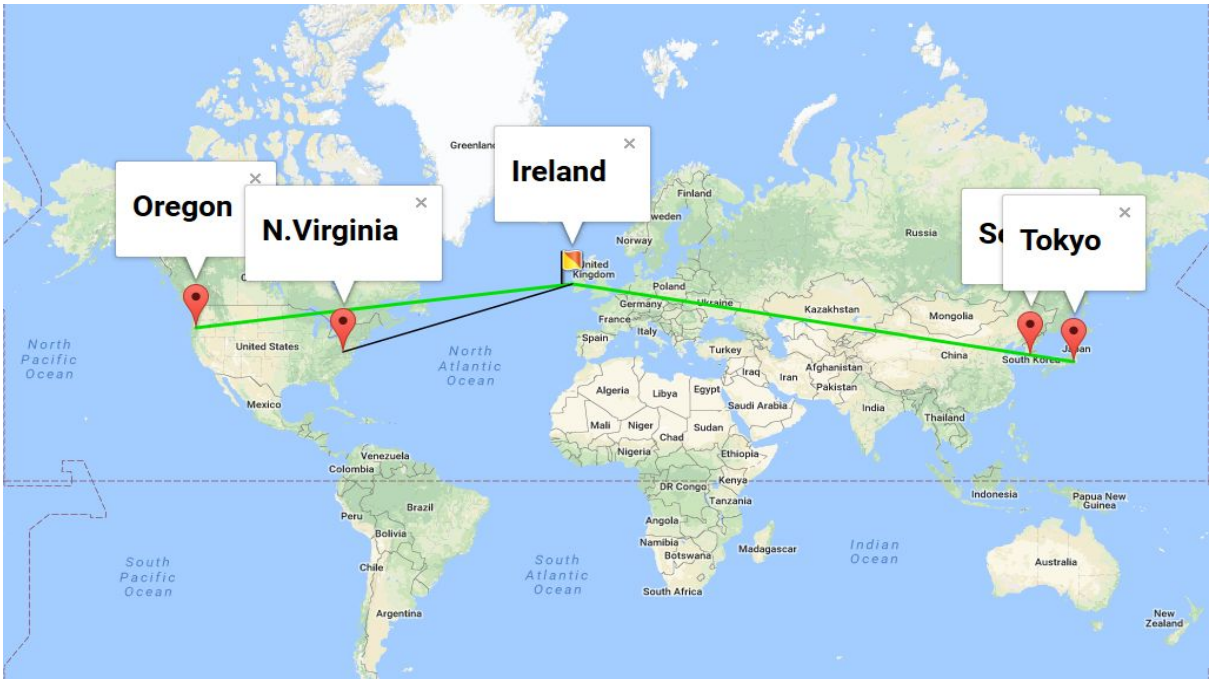


Figure 35: early stage prototype of the IVS using Google Maps.

The nodes's locations of the map above are extracted from EndNodes table found in our database, as shown above in figure 2 found the in background section.
Paths have been chosen randomly, for testing purposes. Furthermore, we decided to modify how the nodes and paths are going to be displayed on the world map, as it is more appropriate to represent the nodes as circles instead of pins, and arrows instead of lines. To achieve that, we have used the Circle and Arrow classes provided by Google Maps API [36].

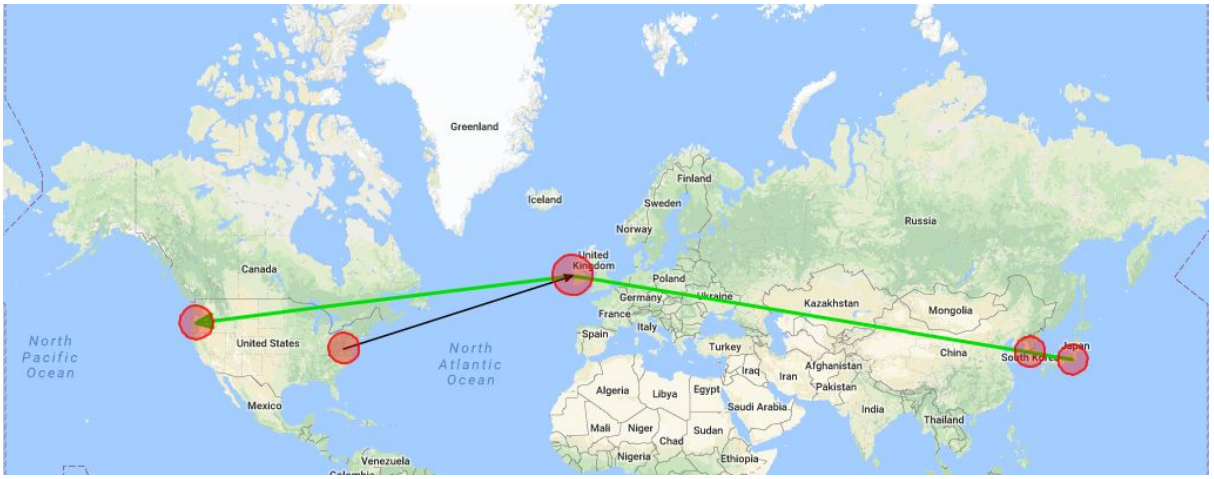


Figure 36: Google Maps-based IVS prototype with arrows and circles

In order to make sure that user interaction can be implemented, we make a sample UI consisting of a form containing of a drop down menu and a checkbox to display all datacenters at once, as shown in the figure 22. Whenever the user interacts with one of the form's element, this triggers HTML events and the map is automatically updated according to user input, thanks to references to the circles and paths that are kept into arrays. In this figure, a sample pair of datacenters is displayed (Northern Virginia–Ireland), with the (here, only one) path joining them.

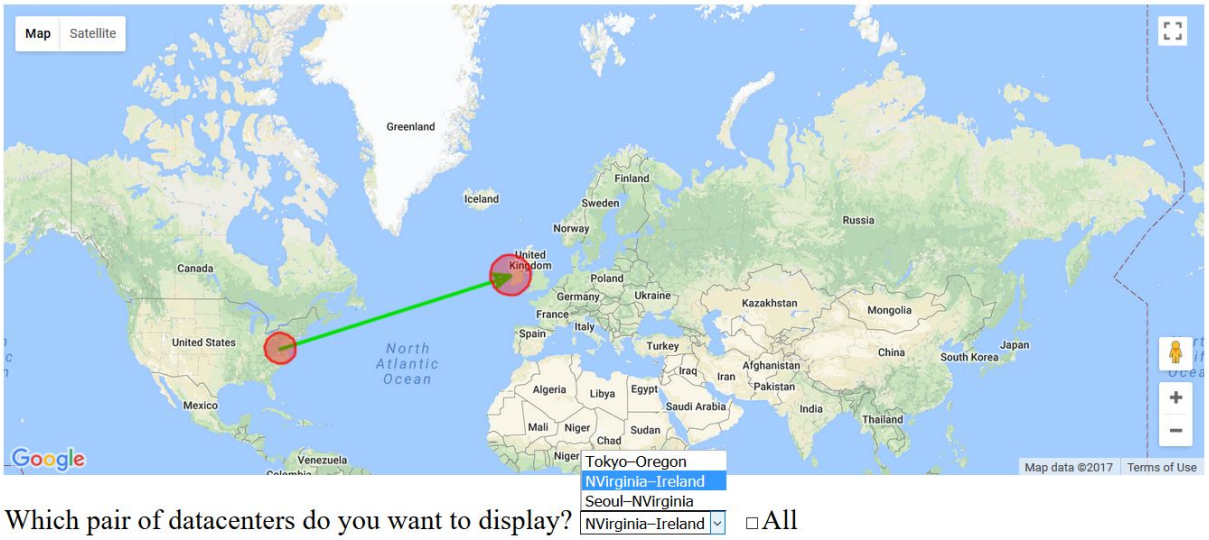


Figure 37: Google Maps-based IVS prototype, with an embedded UI

4.2.2.2 Startup risk of IVS development

After accomplishing 1.3 “Extend measurement database with geographical information”, we draw all idhops on Google Maps. Here is a detailed description on how to retrieve Geolocation related data from MySql to Google map:
Step 1: Create config.php to indicate the database in my local laptop
Step 2: Convert AllgeoHops.csv to AllgeoHops.xml by using output_xml.php
Step 3: Draw the map with index.html which require AllgeoHops.xml to retrieve data.[1]

However, it turns out all more than 50000 Hops map into no more than 10 markers. The cause of this usage lies in the geocoordinate lower-precision problem from MaxMind and the provided dataset. To solve this problem, we checked some others GeoLocation DB, most of which can map IP address to the country, region, city, latitude/longitude. But these geo coordinates are all indicate the center of that city, rather than precise to street address/building number etc.

Analysis on this phenomenon :
As we all know, IP addresses are reutilized and keep updating in a local area network. These register IP addresses are tracked by network provider or related organization. Thus, if we want to locate IP in a more concrete address , we must retrieve the information in real time from their DB. However, they won't open source these informations to public because of network security considerations. It should be reasonable for them to hide the exact location information.

Conclusion:

[1] Note: all the mentioned file can be found from WAN github repository.

We couldn't find an appropriate IP to Geolocation DB to meet our specific requirement and need to chose other plan instead.

4.2.2.3 Proposal for resolving the issue

1. Choose a set of data centers that will used for final evaluation.
2. By looking into the map of the Internet cabling around the world, we find geographic coordinates of the network cables, which allow us to draw these cables on Google Maps[37].
3. Select at least one cable for each pair of datacenters (from the set of four).
4. Once we have the cables placed on the google maps, we should place IP hops on these cables with same distance along the path from src to the dst datacenter.
5. Constructed graph can then be used for our final evaluation.

Deployment on this proposal:
Step 1: Request Geocoordinate informations from Google Elevation API:
Running "request_sample_LatLng_command" for getting a test.xml file with 20 LatLngs.

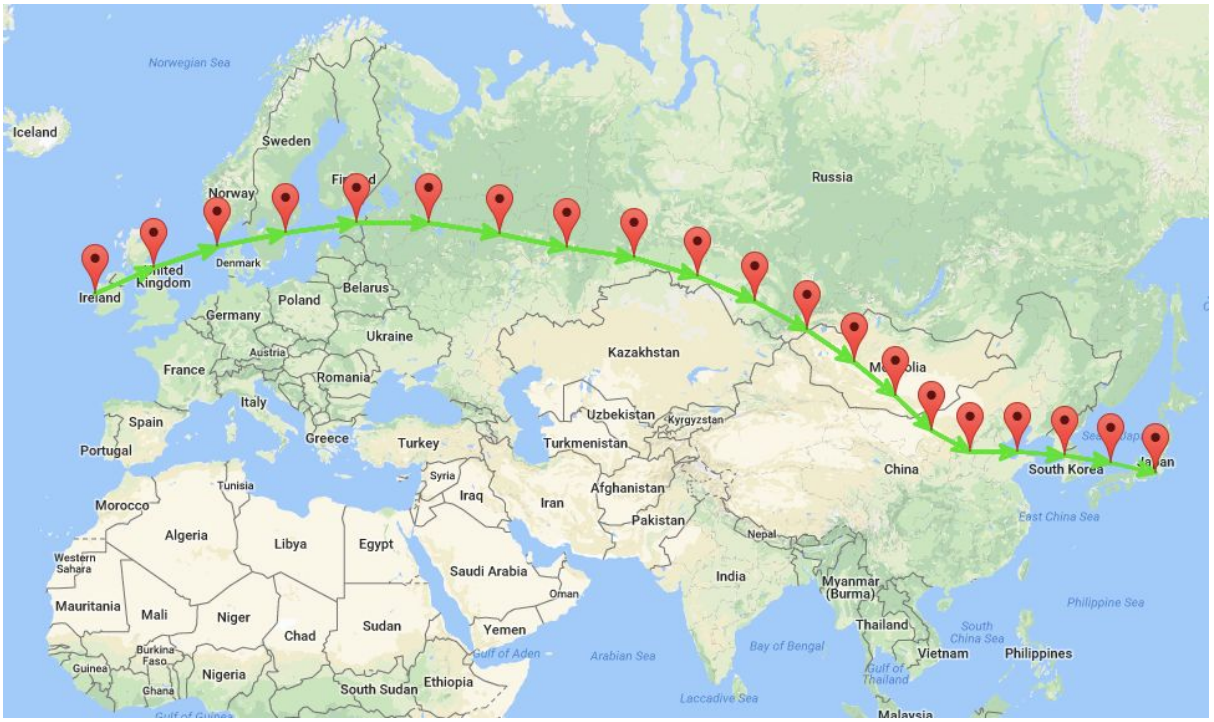


Figure 38: Sample results by using Google Elevation API

Step 2: Merge them into backend database:
Load this file into MySql, following the commands in "load_xml_command.sql".
Step 3: Retrieve the final table which include IP address, IdHops, etc for drawing IVS map:
Retrieve these data by using PHP,including "config.php" and "output_xml.php"
Step 4: Adding all IPHops with equal distance on Google map:
Draw these points on Google Map with "index.html", where we request the results from "output_xml.php".[2]

[2] Note: all the mentioned file can be found from WAN github repository.

Results showing (take instance of all 22 paths from Tokyo to Oregon with equal distance):
At the early stage, we shaped all paths into submarine cables.

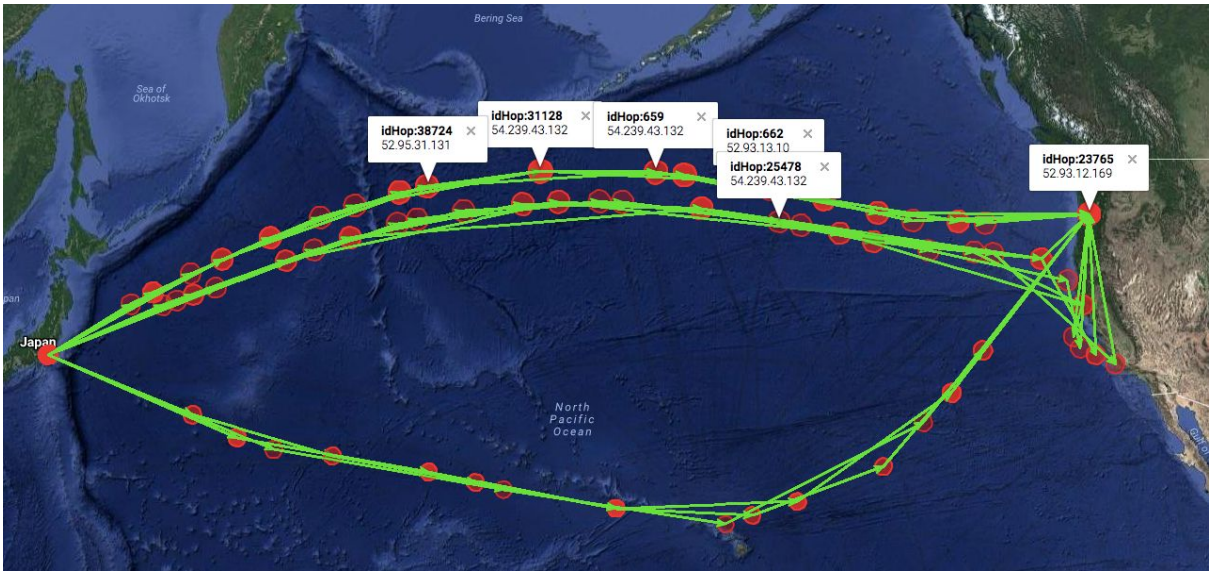


Figure 39: Three Submarine cables from Tokyo to Oregon

Aiming to make it more practicable, we shaped all paths into terrestrial cables. Then we add an interaction board to choose different cable.

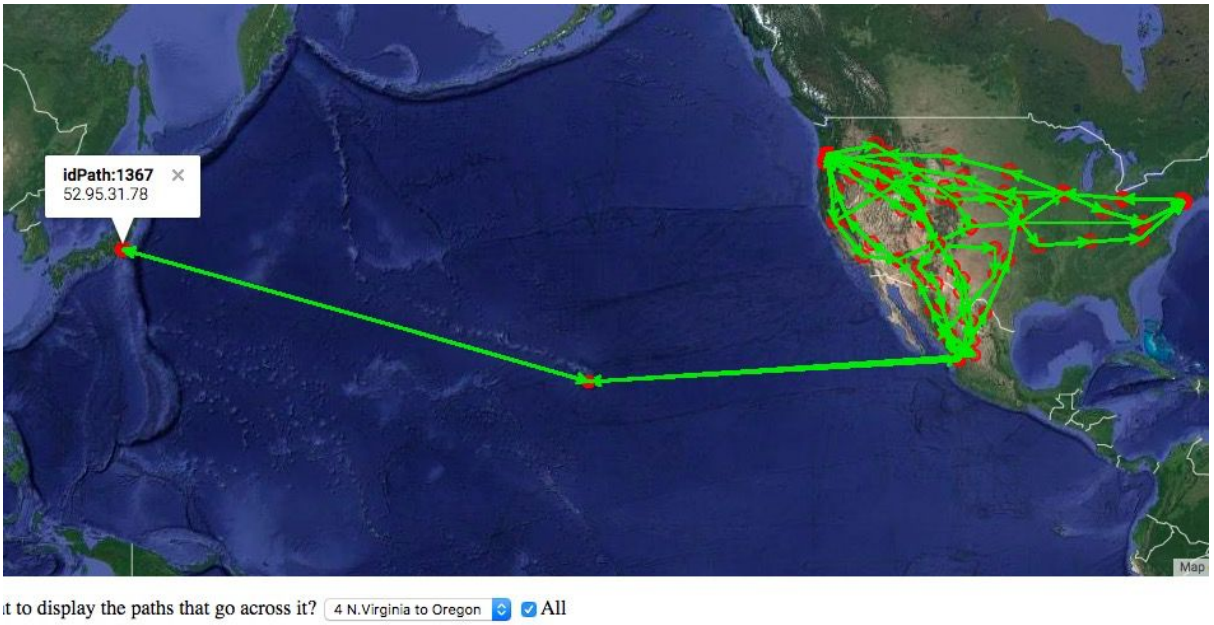


Figure 40: Terrestrial cables from Tokyo to USA

As shown in Figure 39 and Figure 40, different paths belong to the same cable cannot coincide exactly, because we don't have exact GeoCoordinate for each ip. To be more specific, Google Elevation API can help us to sample points in a line. We need to request

Elevation API with GeoCoordinates of two terminal points and turning points in this line. Then it reply with GeoCoordinates of every sample points.
So, we indicate 3 directions for 3 polylines firstly (this part is manual and follow the real submarine cable), then indicate which path belong to which cable. Because each path has different amounts of ipHops, they cannot be distributed in the same geographic points. At last, 22 paths can pass with existing 3 directions.

The result showing cables between Oregon and N.Virginia is more readable because of few amount of paths:

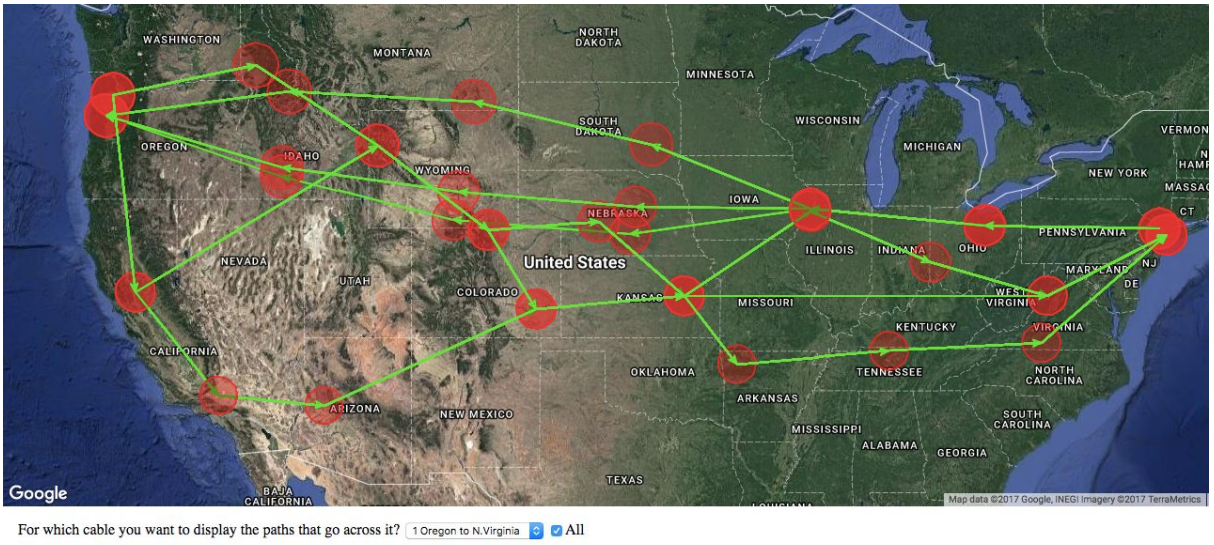


Figure 41: Bidirectional cables in USA

4.2.2.4 Design IVS as real word network topologies

To modify the original proposal, we finally developed the IVS based on real word network topologies [38]. Specifically, we picked up five different countries then download internet graphs of them that are already encoded in GML format [39]. Then showing these nodes on Google Maps by converting GML to JSON, which can be used as a basic data format in when developing the expert system. Lastly, we added an interaction board to make a selection among five different countries. The figure below shows the results of a network topology for Germany:

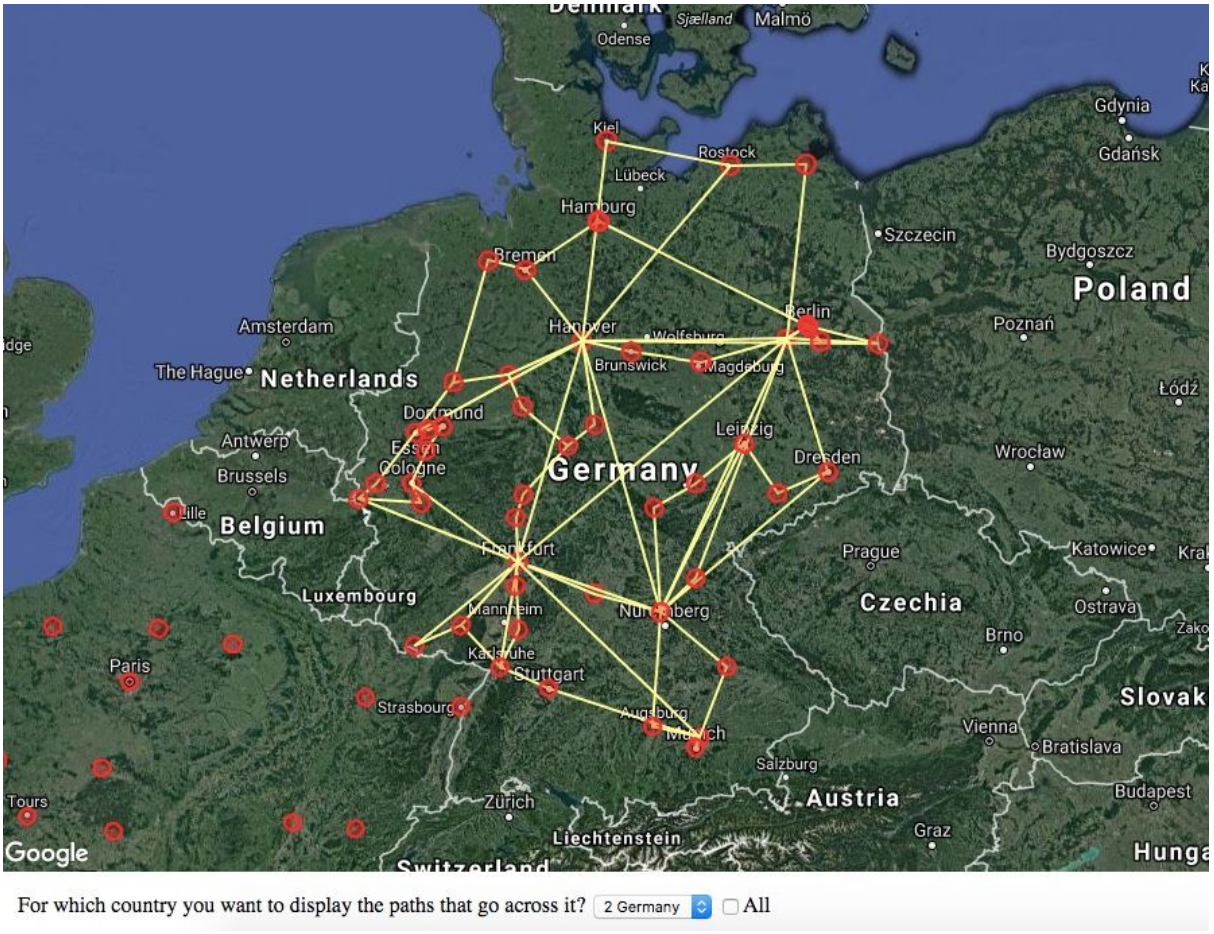


Figure 42: Internet topology in Germany

4.3 Design and develop Expert Infrastructure Change System (EICS)

In this section, we will develop an expert system capable of proposing alternative connections (routes) on the graph based on the available information and available resources. but before fully designing our EICS, we started by replacing our network topology with a new and real world network topologies provided by The University of Adelaide [39]. Our new topology is in a JSON format and are divided into a two JSON arrays, one for nodes and other for links. Nodes contains ID for a city and its latitude and longitude. Links contains a link between one source and one destination city. The figure below shows in details how the given JSON file looks like.

```
"nodes": [
{
"Latitude": 36.17497,
"Country": "United States",
"Internal": 1,
"id": "Las Vegas",
"num": 1,
"Longitude": -115.13722
},....]
```

```
"links": [
{
"LinkType": "DS-3",
"source": "Las Vegas",
"snum":1,
"LinkLabel": "45 Mbps DS-3",
"LinkNote": "45 Mbps ",
"target": "Phoenix",
"tnum":12
},....]
```


Figure 29: JSON format that contains “nodes” and “Links”

The first step in implementing the EICS was to calculate the shortest path using Dijkstra’s algorithm and display this path on the world map. This required reading the JSON file, calculate the geographic distance using a node’s latitude and longitude, create a graph using source and destination latitude and longitude and finally run Dijkstra’s algorithm.

1. Reading JSON file

This required to add extra libraries to the Java integrated development environment (IDE), e.g IntelliJ or Eclipse. Afterwards, start reading the JSON file by using a JSON parser and save the data as JSON objects. Moreover, we have created two JSON arrays to save the nodes and links from the file.

2. Calculating the geographic distance

In order to calculate the distance between two nodes, we make the assumption that the earth is a spherical object (thus ignoring ellipsoidal effects), which can lead to errors up to 0.3%.[40] We are using the great-circle distance or orthodromic distance, which is the shortest distance between two geographical points on the surface of a sphere (in this case the earth). The distance between the two points is the length of a straight line between the two points. The formula is using the spherical law of cosines which takes the geographical coordinates in radians.
The formula that is being used is as follows:

$$d = arccos(sin\phi 1 * sin\phi 2 + cos\phi 1 * cos\phi 2 * cos(\Delta\lambda)) .$$

These two points are $\phi 1$ and $\phi 2$ are the latitudes of the first and second point respectively, while the $\Delta\lambda$ is the difference between the longitudes of the two points. Afterwards, we convert the value that we get from the formula to radians and then multiply it with (60 * 1.1515) where 60 is the number of minutes in a degree and 1.1515 is the number of statute miles in a nautical mile (1NM= 1.1515 statue miles). Then we multiply this value with 1.609344 which is the number of kilometers in a mile.

3. Creating the graph

The graph is created by using the source and destination nodes found in links and the distance between these nodes. Graph will contain nodes that will be assigned neighbouring nodes given the set of links provided. These nodes are later used in the Dijkstra function.

4. Create a server

We have created a server that will enable the communication between the frontend and the backend, i.e. between the IVS and the EICS. To run our shortest-path algorithm, we start by running the server, which will enable us to load our visualisation tool by opening a browser and typing “localhost:4567” in the address bar. Then the user can choose which topology to run Dijkstra on and the server will return the shortest path that the IVS will display on the world map.

5. Run Dijkstra’s algorithm

After the above steps have been implemented, the algorithm will find the shortest path according to the distance between a given source node and destination node. The result is then sent to the front end as as a JSONArray to be displayed on the world map. The user can either choose to select and specific source and destination nodes for the algorithm to run or can run the algorithm to find the shortest path for every source and destination nodes.

After the previous steps have been implemented, we started creating the EICS. Creating this system requires the following steps to be achieved:

1. Different approaches for implementing the EICS algorithm.

We started by developing a pseudo code of how the EICS should work together with Dijkstra’s algorithm. The algorithm will propose better paths to be taken when running Dijkstra’s algorithm. The algorithm will find nodes and link these nodes to the existing nodes on the world map that would be the shortest path from a given source and destination node. We have thought of two approaches to implement the EICS, but we have used only one of them due to some time constraints. The first approach is a brute-force searching and second one is circle algorithm.

a. Brute-force searching:

Brute-force search is a very general problem-solving technique that consists of enumerating all possible candidates for the solution and checking which candidate satisfies the problem's statement. Although the brute-force search is simple to implement, and will always find a solution. Its complexity is proportional to the number of candidates, which leads to many practical problems: the cost tends to grow very quickly at the same time with the size of the mesh increases in density. Here we need to generate a fixed number of nodes in the graph we already have. Each of these nodes is placed with a fixed distance between each other. Then give links to all of them, try to run Dijkstra again and find a more optimized solution.
We started writing a simple pseudocode of the EICS algorithm. This pseudocode will help us to define what are the important parts of the algorithm that needs to be included before the actual implementation. The figure below shows the developed pseudocode.

```
Run Dijkstra's algorithm for specified source node
Store the cost to the target as the best distance
For all new nodes
        Add node Nx to the Graph
        For all pair of nodes that existed in the initial graph
          Add the two new nodes as neighbors of Nx if constraint is met
          Run Dijkstra's algorithm for the specified source node
          Get the distance to the target node (cost of the target)
          Remember this node and its neighbors if the distance is better.
```

Figure 46: Pseudocode for EICS

b. Circle algorithm:

The algorithm was designed with a set of requirements to be implemented. The first step in the algorithm is to check if there is a possibility to add a link between the given source and destination, if the constraint allows to create such a link. Otherwise, draw a circle using the source node as the center of the circle whose radius is the constraint. When the circle intersect with any link, then we create a node (intersection point) and make a link between the source node and the intersection point. Then run Dijkstra's algorithm to find the shortest path between source and destination using this intersection point. This process is repeated for every node in the current shortest path. The figure below shows a detailed description of how the actual algorithm works.

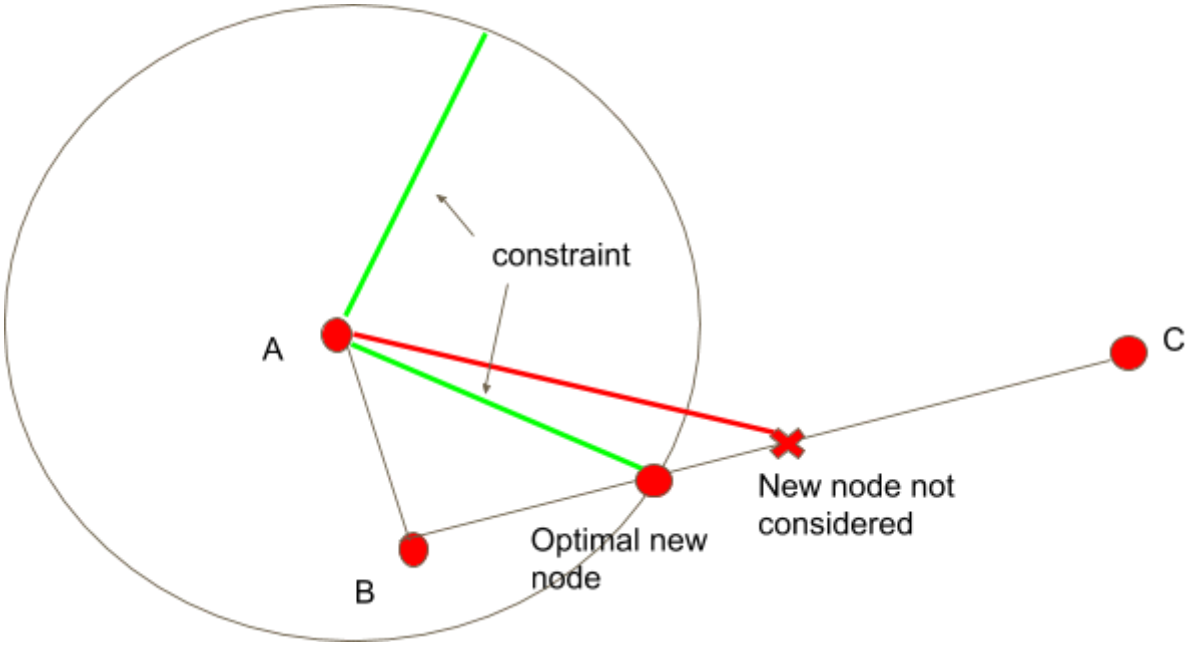


Figure 43:

Figure 29: Circle algorithm

We chose to go with the first approach for developing the algorithm of the EICS. Since it can be used with any network topology and easier to perform performance evaluation.

2. Create a new meshes.

The first step in developing the algorithm is to generate new JSON files that contain multiple nodes with geographic information. Each node is placed with a fixed distance between each other. We made different topologies for different countries (Finland, Germany, USA, China, and France) and different densities. By the density, we mean that the topology contains a different number of nodes or different distances between the nodes. These nodes will be used together with our earlier/current topologies to help propose new nodes and links. The figures below show samples of the new JSON file used for the EICS.

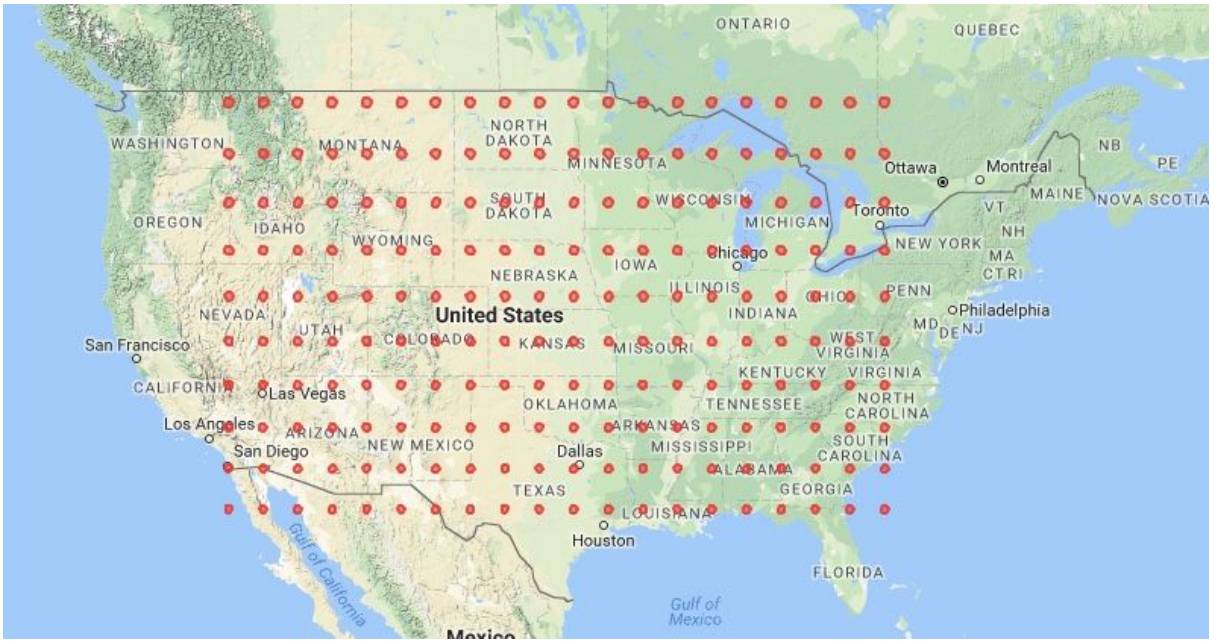


Figure 44: USA map with geographic information for the EICS.

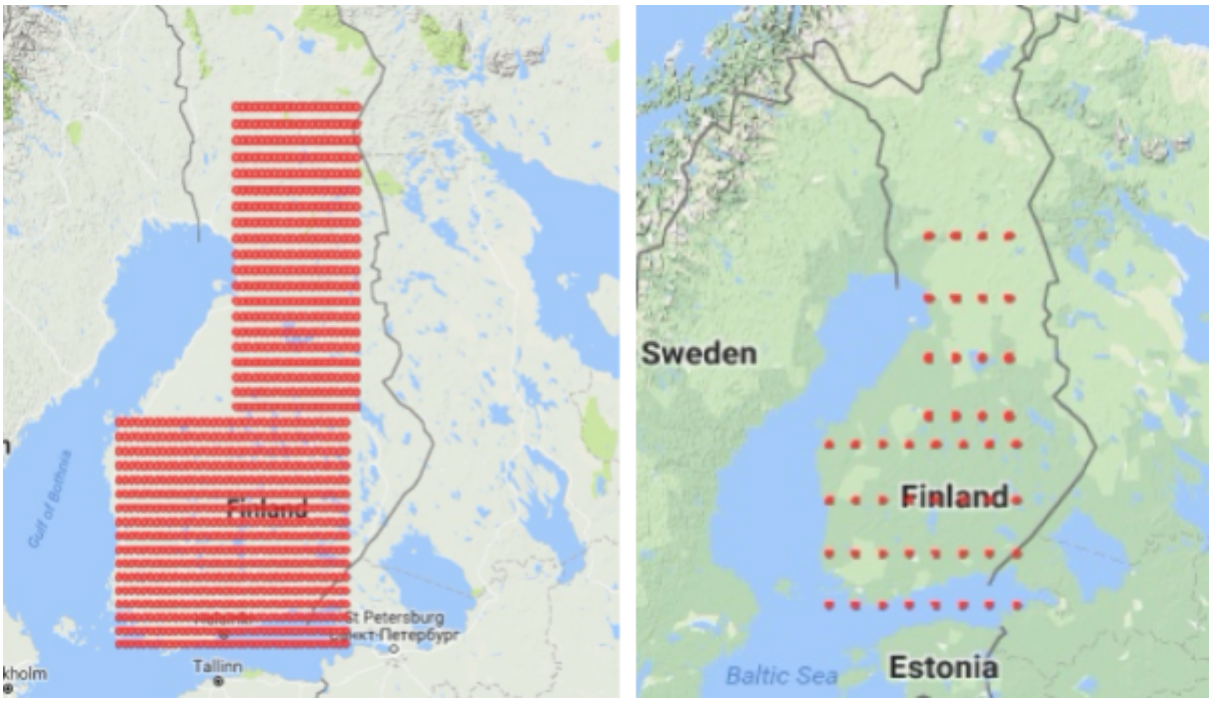


Figure 45: Finland map with different node densities and distances.

The reason of why we chose to pre-generate meshes instead of programmatically creating meshes when the algorithm is running. The pre-generation approach will enable the user of the system to avoid placing network nodes(hops) on mountains and in water.

3. Workflow of EICS

The user will interact with the visualization tool that contains GUI elements to display the shortest path given a source and destination node, zoom functionality for any network topology, and request new topology. The backend part contains the server and EICS. The server will handle the request sent from the frontend by the IVS, request topology improvement from the EICS, and the EICS will return an improved topology to the server and back to the IVS. The figure below shows how the IVS is communicating back and forth information with the EICS.

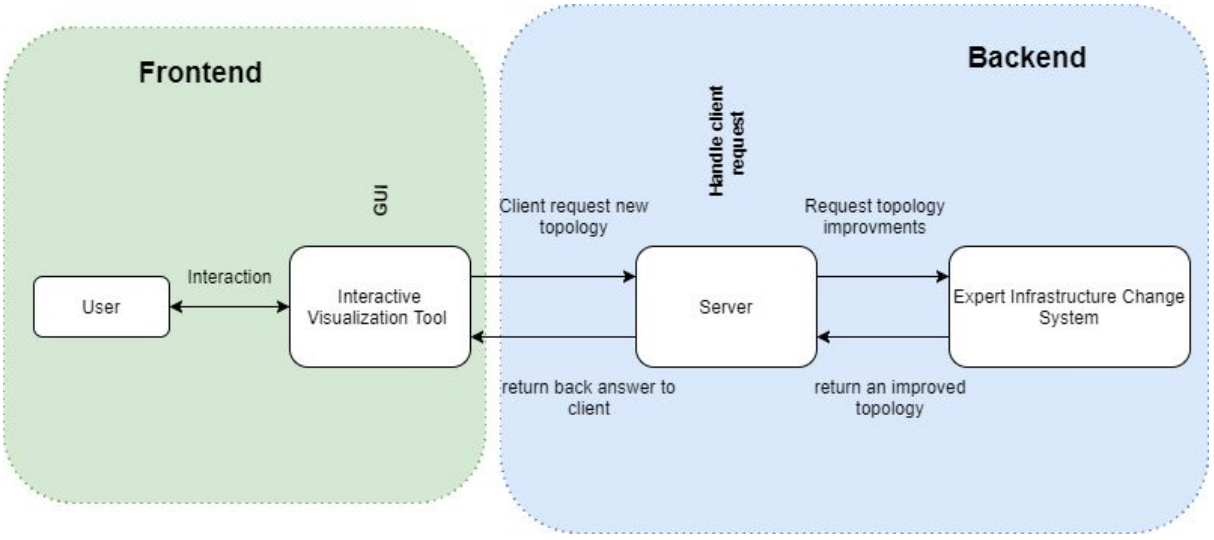


Figure 47: Workflow of EICS with IVS

4. Complexity of EICS

The complexity of the EICS algorithm is $O(N^2 * (V\ choose\ 2) * E * log(V))$, Where $E * log(V)$ is the complexity of the Dijkstra's algorithm and N^2 is the size of the mesh. since we are adding one vertex and two links(edges) we can replace E with $(E+2)$ and V with $(V+1)$.

Benefits of using this EICS:

1. **Manipulate and interact directly with network data**. One of the greatest strengths of data visualization is how it brings actionable insights to the surface. Unlike one-dimensional tables and charts that can only be viewed, our IVS (Interactive Visualisation System) enable network administrators to interact with data. For example, an administrator cannot realize the new route has placed in a zone of geographical barrier (e.g. ocean or sea). But it's much directly to find this issue if we found it on the visualisation tool

2. **Introduce new hops to existing topologies with required link distance limit.** Link length (cable) is also a cost consideration. if the cable is too long, we should also consider choose a new route nearby to have a best balancing cost performance.
3. **Identify and act on emerging trends faster.** Using data visualization is its ability to tell a whole network topology through data. Administrators could see the traffic flow firsthand.

4. The knowledge base (rules) could be updated and extended, it's much more flexible and efficient.

Limits of the EICS:

1. The Brute-force searching algorithm does solve the problem but is not so effective.
2. The EICS is not able to learn from the mistakes (for example geographical barriers), no common sense used in making decisions.
3. We do not consider balancing network traffic (it should be considered in real life if we sale EICS to network providers).

Performance Evaluation

In this section, we evaluate the performance of the algorithm using different topologies. We started our evaluation by selection one topology at a time and run at for ten times. The evaluation was also done by changing the values of the topologies' densities and link lengths(constraint or cost). The performance values in time in milliseconds were obtained by taking the median of times of running the topologies. The tables below show the result obtained after evaluating two countries (Germany and Finland) with different densities and link lengths.

Table 4: Running time for Germany with regards to different density and cost.

	Density		
Cost in KM	Low	Middle	High
100 km	18 ms	44.3 ms	92.2 ms
350 km	131.1 ms	709.2 ms	1229. ms
1000 km	663.4ms	3852.7 ms	8319.2 ms

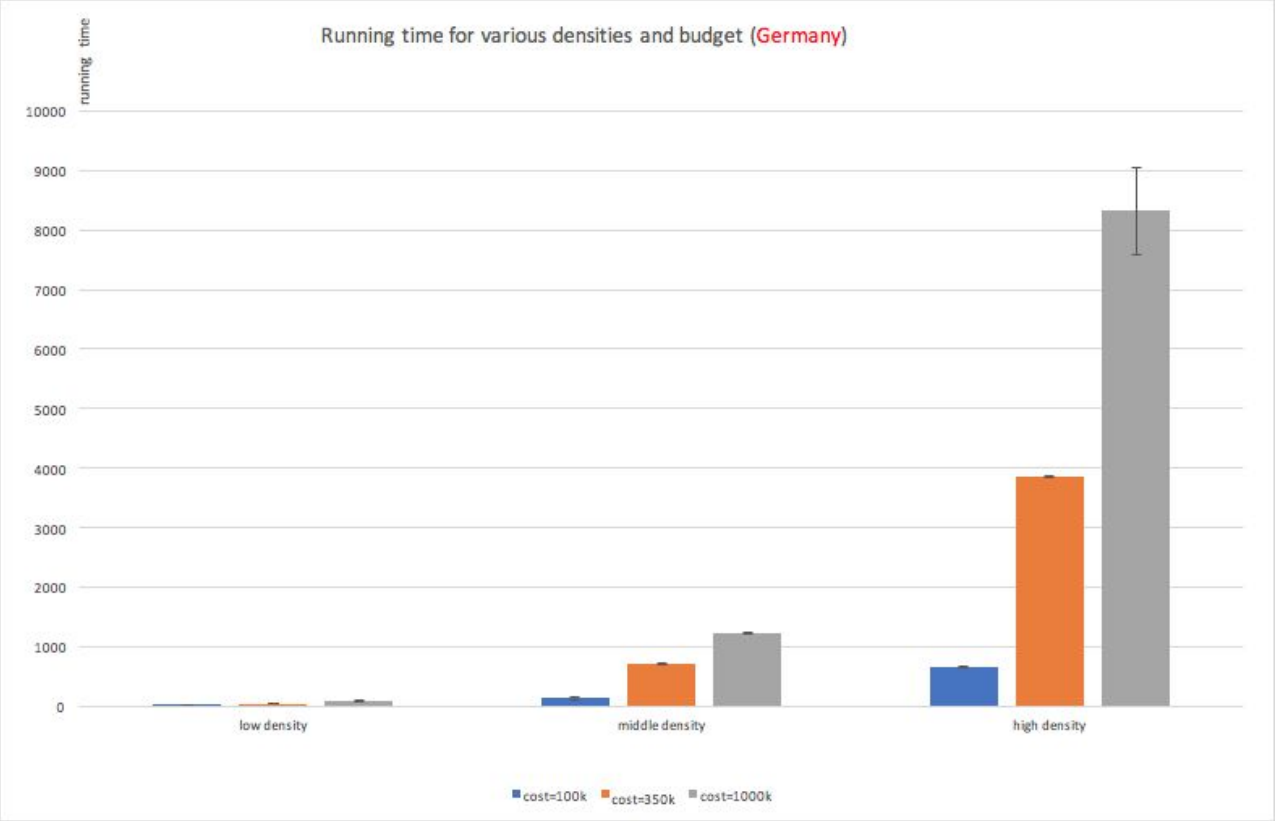


Figure 46: running time for various densities and budget (Germany)

Table 5: Running time for Finland with regards to different density and cost.

	Density		
Cost in KM	Low	Middle	High
110 km	2.8 ms	15.4ms	67.8 ms
250 km	10.2 ms	61.5 ms	202 ms
700 km	57.4 ms	303.1 ms	1222.2 ms

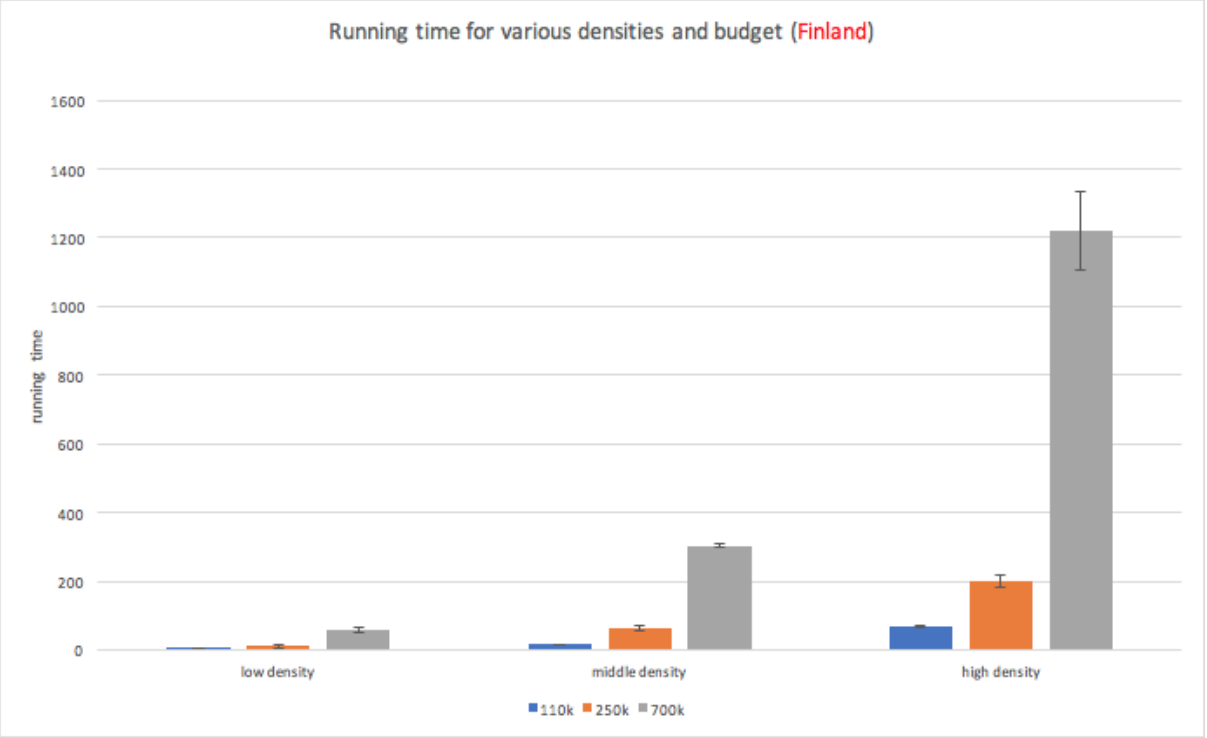


Figure 47: Running time for various densities and budget (Finland)

The cost of links for Finland's topology is different than what is calculated for Germany's topology. That differences in costs are due to the fact to the size of Finland's topology is smaller than Germany's topology.

Conclusion

Future Work

There are lots of areas of improvement for the EICS algorithm. The fact is that we are testing with one node at a time is not the most optimal solution and the obtained values in the performance evaluation could has been more interesting. The algorithm needs to add more than one node at a time. There are also other factors that we think that could be improved. These factors are as follows:

1. Find a better algorithm to reduce the total distance (latency) more efficiently. For example, add more nodes to the algorithm instead of having only one. Use the circle algorithm to test.
2. We do not consider balancing network traffic, it should be considered in real life if we sale EICS to network providers. We need to employ load balance to enhance reliability and increase resource utilization.
3. Our EICS is not able to learn from the mistakes (for example geographical barriers), no common sense used in making decisions. We would use machine learning to help the system to avoid geographical barrier zones automatically.

References

[1] K. Bogdanov, “Interactive Visualisation System for Wan Paths,” 2017.

[2]“MaxMind,” *GitHub*. [Online]. Available: https://github.com/maxmind. [Accessed: 04-Jan-2018].https://www.maxmind.com/en/home. [Accessed: 03-Oct-2017].

[3] “Adding a Google Map with a Marker to Your Website | Google Maps JavaScript API,” *Google Developers*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/adding-a-google-map. [Accessed: 04-Jan-2018].)

[4]“Getting Started | Google Maps Android API,” *Google Developers*. [Online]. Available: https://developers.google.com/maps/documentation/android-api/start. [Accessed: 04-Jan-2018].

[5] “Introduction to the Google Maps Android API | Google Maps Android API,” *Google Developers*. [Online]. Available: https://developers.google.com/maps/documentation/android-api/intro. [Accessed: 04-Jan-2018]

[6]“About,” *Gephi makes graphs handy*. [Online]. Available: https://gephi.org/about/. [Accessed: 04-Jan-2018].

[7] “Gephi - Editor Review, User Reviews, Features, Pricing and Comparison in 2017,” *Predictive Analytics Today*, 01-Oct-2016.

[8] “review | Gephi blog,” *Scientist Christian Tominski about Gephi*. .

[9] “Developers,” *Mapbox*. [Online]. Available: /developers/. [Accessed: 04-Jan-2018].

[10] A. G. M. G. M. Science, “A Powerful New Mapmaking Tool Fit for Both Pros and Newbies,” *WIRED*. [Online]. Available: https://www.wired.com/2015/09/mapbox-studio/. [Accessed: 04-Jan-2018].

[11] M. Myerson, “MapBox: Straightforward Beautiful Map-Making « Web.AppStorm,” 30-Oct-2013. [Online]. Available: http://web.appstorm.net/reviews/media-reviews/mapbox-straightforward-beautiful-map-making/. [Accessed: 04-Jan-2018].

[12] S. Shakkottai, R. Srikant, N. Brownlee, A. Broido, and K. claffy, “The RTT Distribution of TCP Flows in the Internet and its Impact on TCP-based Flow Control.”

[13] Kuhn, Nicolas, and David Ros. “Improving PIE’s Performance over High-Delay Paths.” *ArXiv:1602.00569 [Cs]*, February 1, 2016. http://arxiv.org/abs/1602.00569.

[14] “What Is Round-Trip Time (RTT)? - Definition from WhatIs.Com.” SearchNetworking. Accessed November 7, 2017. http://searchnetworking.techtarget.com/definition/round-trip-time.

[15] “使用jmeter测试java程序包性能--20130111总结liwf.” [Online]. Available: https://www.testwo.com/blog/6334. [Accessed: 04-Jan-2018].

[16] Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs,". Numerische Mathematik 1: 269–271. doi:10.1007/BF01386390

[17] A. Crauser, K. Mehlhorn, U. Meyer, P. Sanders, “A parallelization of Dijikstra’s shortest path algorithm”, in Proc. of MFCS’98, pp. 722-731, 1998.

[18] V. Beal, “What is Latency? Webopedia Definition.” [Online]. Available: https://www.webopedia.com/TERM/L/latency.html. [Accessed: 04-Jan-2018].
[19] “What is Network Latency and Why Does It Matter?” [Online]. Available:https://www.o3bnetworks.com/wp-content/uploads/2015/02/white-paper_latency-matters.pdf

[20] Lisa M. (2008). The Sage encyclopedia of qualitative research methods. Los Angeles, Calif.

[21] Babbie, Earl R. The Practice of Social Research. 12th ed. Belmont, CA: Wadsworth Cengage, 2010).

[22] Denzin, Norman K.; Lincoln, Yvonna S., eds. (2005). The Sage Handbook of Qualitative Research (3rd ed.).

[23] “IP Geolocation and Online Fraud Prevention | MaxMind.” [Online]. Available: https://www.maxmind.com/en/home. [Accessed: 03-Oct-2017].
[24] M. Pablo, “Using the MaxMind GeoLite City data with MySQL.” [Online]. Available: http://odkq.com/geolitecity. [Accessed: 03-Oct-2017]) .
[25] “GeoIP Legacy CSV Databases « MaxMind Developer Site,” *MAXMIND*. [Online]. Available: http://dev.maxmind.com/geoip/legacy/csv/. [Accessed: 05-Oct-2017].
[26] “Instant IP Address Lookup,” *WhatIsMyIPAddress.com*. [Online]. Available: http://whatismyipaddress.com/ip-lookup. [Accessed: 05-Oct-2017].
[27] “Gephi - The Open Graph Viz Platform.” [Online]. Available: https://gephi.org/. [Accessed: 04-Jan-2018].
[28] A. Jacomy, ‘GeoLayout’, 24-Sep-2017. [Online]. Available: https://gephi.org/plugins/#/plugin/geolayout-plugin. [Accessed: 10-Oct-2017].
[29] “Features.” [Online]. Available: https://gephi.org/features/. [Accessed: 05-Jan-2018].
[30] “Google Maps APIs,” *Google Developers*. [Online]. Available: https://developers.google.com/maps/. [Accessed: 05-Jan-2018].
[31] “Arrows (Symbols) | Google Maps JavaScript API,” *Google Developers*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/examples/overlay-symbol-arrow?hl=zh-cn. [Accessed: 05-Jan-2018].
[32] “The Google Maps Geolocation API | Google Maps Geolocation API,” *Google Developers*. [Online]. Available: https://developers.google.com/maps/documentation/geolocation/intro. [Accessed: 05-Jan-2018].
[33] “Custom Marker Symbols | Google Maps JavaScript API,” *Google Developers*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/examples/marker-symbol-custom. [Accessed: 05-Jan-2018].
[34] “Using MySQL and PHP with Google Maps | Google Maps JavaScript API,” *Google Developers*. [Online]. Available:

https://developers.google.com/maps/documentation/javascript/mysql-to-maps. [Accessed: 05-Jan-2018].
[35]“Mapbox API Documentation.” [Online]. Available: https://www.mapbox.com/api-documentation/#geocoding. [Accessed: 05-Jan-2018].
[36] “Google Maps JavaScript API V3 Reference | Google Maps JavaScript API,” *Google Developers*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/3.exp/reference. [Accessed: 05-Jan-2018].
[37] “Submarine Cable Map,” *https://www.submarinecablemap.com/*. [Online]. Available: https://www.submarinecablemap.com/. [Accessed: 06-Jan-2018].
[38]“The Internet Topology Zoo.” [Online]. Available: http://www.topology-zoo.org/explore.html. [Accessed: 06-Jan-2018].
[39] “The Internet Topology Zoo.” [Online]. Available: http://www.topology-zoo.org/dataset.html. [Accessed: 06-Jan-2018].
[40] whuber, “How accurate is approximating the Earth as a sphere?,” *StackExchange*. 16-May-2012.

Appendix 1

The frequency of observing new paths (Subsection 1.1 Demonstrate that the number of network paths between any two datacenters is finite)

The finite number of network paths can be demonstrated via analysis of the frequency of observing new paths between any 2 sampled endpoints (i.e., source and destination data centers).

This is the pdf and cdf data that was used to get all average persistence of network paths for subsection 1.2 Analyse persistence of network paths.