Software Design Document
for
Network Simulator

Version 2.0 approved

Prepared by:
Faculty Advisor:
Zilong Ye
Team Leader:
Andy Do
Team Members:
William Fong
Daniel Romo
Y Hoang
Zifan Yang
Dibakar Barua

Sponsors: Cal State LA CS Department

April 14, 2018

Revision History

Name	Date	Reason For Changes	Version
First Boot	10/12/17	Initial Changes on section 1 and section 10	1.0
Final	4/14/17	Final changes to the revision	2.0

1. Introduction

1.1 Purpose

The Network Simulator, also known as NS from this point forward, will allow any user to evaluate various networking algorithms that are built into the software.

1.2 Document Conventions

This document follows the MLA Format. Bold-face text has been used to emphasize section and sub-section heading. Highlighting is to point out words in the glossary and italicized text is used to label and recognize diagrams.

1.3 Intended Audience and Reading Suggestions

This document is intended for developers and testers. Developers can look through this document to understand how the software works and how the modules communicate with each other. Testers can use this document to find out any errors in the modules in the event of a bug occurring in the software. To fully grasp our project, we recommend reading this document in the following order:

1. 4 - System Architecture
2. 5 - Policies and Tactics
3. 6 - Detailed System Design
4. 7 - Detailed Lower Level Component Design
5. 9 - User Interface

By reading this document in this order, you will be able to understand how this project came together and how it works.

1.4 System Overview

The Network Simulator (NS) will be written in Java and JavaScript. It will provide a user-friendly GUI that users can navigate through. The user will be able to select points on Google Maps and save them into a topology file. Additionally, the user will be able to run a simulator using various routing algorithms to determine which algorithm is the fastest.

2. Design Considerations

2.1 Assumptions and Dependencies

The NS is coded in Java and JavaFX, thus it runs on the Java Runtime Environment (JRE). It shall run on any Windows / Macintosh / Linux computer that has the JRE installed.

2.2 General Constraints

The NS will require a modern computer with a powerful processor to be able to handle all the simulations.

2.3 Goals and Guidelines

The goal of NS is to make an application that is user friendly. It should be able to simulate the various routing algorithms developed without any issues.

2.4 Development Methods

Throughout the process of development, we will be following the Agile Development model. Every week, we will meet to exchange our progress that was done in the previous week. To ensure that we accomplish our weekly goal, we will split the work among ourselves. We will check the progress among ourselves daily via online messaging. This will allow us to develop the NS while maintaining a consistent development cycle.

3. Architectural Strategies

3.1 Use of a particular type of product (programming language, database, library, etc.)

NS is being developed on the Java platform using Eclipse as the integrated development environment (IDE). NS's User Interface(UI) is being developed on the JavaFX. JavaFX is a sub-language derived from the Java programming language for design UI.

3.2 Reuse of existing software components to implement various parts/features of the system

NS is utilizing GMapFX API to display a map in the UI.

3.3 Future plans for extending or enhancing the software

Future plans for NS are including but not limited to:

- Improving the cosmetic look of NS for a more pleasurable viewing experience.
- Developing an offline application that will allow users to access the same functions included in as NS.
- Implementing further support for custom input for NS.

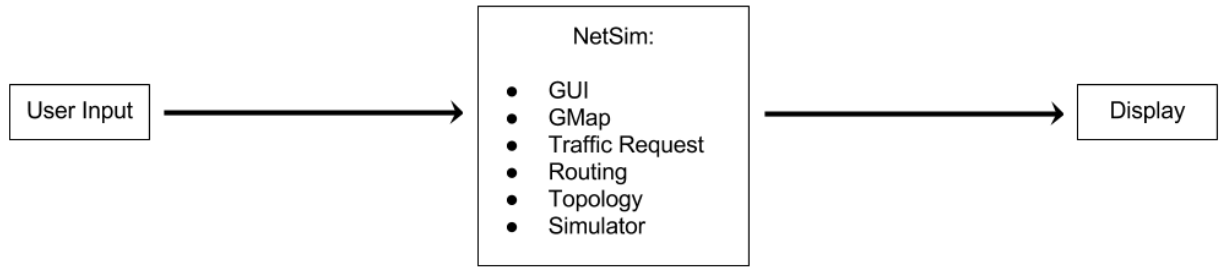
3.4 User interface paradigms (or system input and output models)

Use of NS will be using the graphical display paradigm. Input of data to NS for use in functions will be in user input textboxes, dropdown choice menus or text files in either .csv or .txt. Output of NS will be given via visual chart and graph and text files.

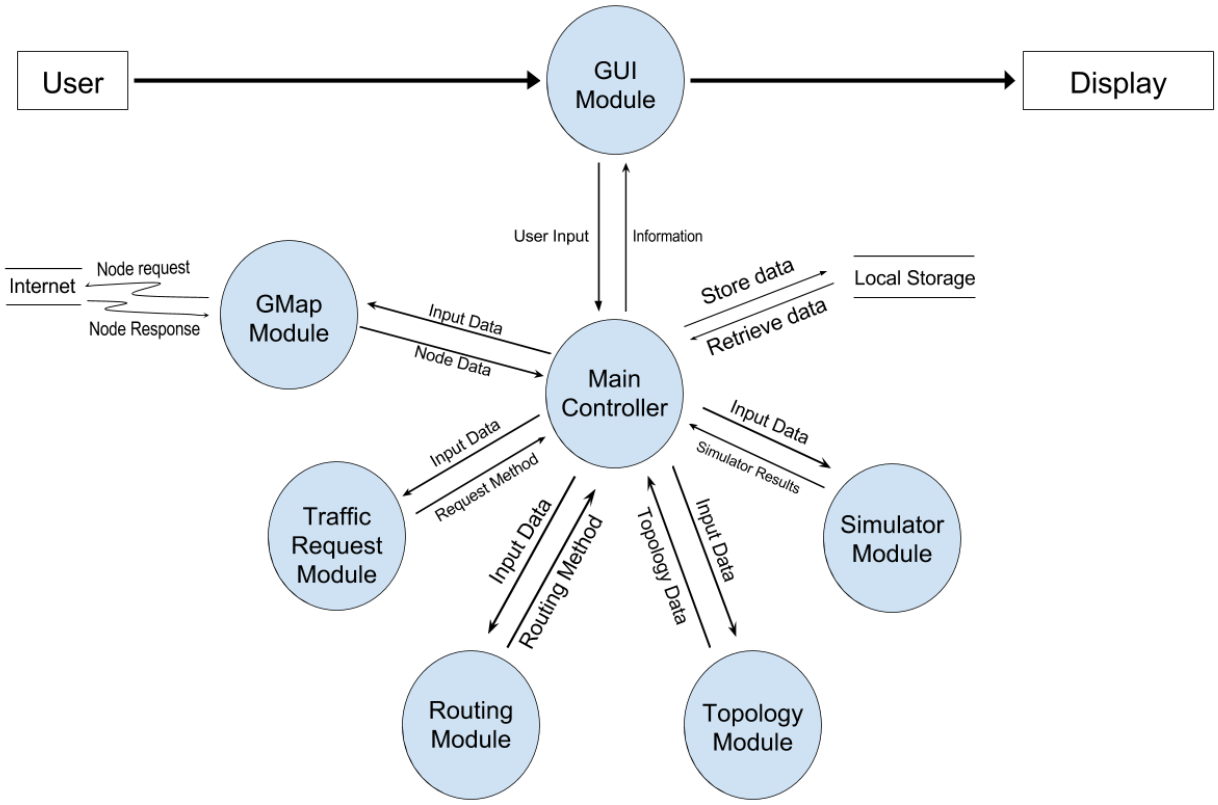
4. System Architecture

The NS has been broken up into sections in order to allow a clear and logical development process.

Level 0 DFD



The level 0 DFD is broken up into three parts. When starting the NS, the user will navigate thru the application and will have the results sent to the display.



Level 1 DFD

Main Controller

- This module will handle the exchange of data between various modules. For example, it will take the routing method selected in the routing module and send it to the simulator module.

GUI

- This module is what allows the user to see the application and interact with it. It will be able to support various display resolutions and sizes that are available in the market today.

GMaps

- This module will allow the GUI to display Google Maps. It will allow the user to create, edit, and disable nodes.

Traffic Request

- This module will allow the user to select between several traffic request methods. This selection will be passed on to the simulator.

Routing

- This module will allow the user to select between several routing methods. This selection will be passed on to the simulator.

Topology

- This module will allow the user to take the nodes that were made in the Google Maps module and save it into a file. In addition, the user will be able to import a custom topology file that contains a custom set of nodes to be displayed on Google Maps.

Simulator

- This module will take the routing method and traffic request method that was selected by the user and run a simulation. At the end of the simulation, it will take the results and display graphs using them.

5. Policies and Tactics

5.1 Choice of which specific products used

Eclipse was used to develop the NS. The GMapsFX API was used to display Google Maps in the GUI. For testing, Windows and Mac computers were used.

5.2 Plans for ensuring requirements traceability

Throughout the development process, the design and requirements document will be frequently updated to match our latest revisions to the software.

5.3 Plans for testing the software

The NS will be tested for functionality and design. To test the functionality of the NS, the simulator will be run to confirm the results. Additionally, test points will be made on Google Maps and verify that the topology is correctly saved. To test the design of the NS, it will be run on several computers with to ensure the GUI fits in the screen.

6. Detailed System Design

Section 6: Detailed System Design

6.1 module-1 Main

6.1.1 **-** The Main Module (MM) will provide the ability to link data and other interactions between all other modules in the diagram.

6.2 module-2 User Interface

6.2.1 **-** The User Interface Module (UIM) is responsible for receiving user input and then displaying any of the information given in an easy to look at interface. The (UIM) is to develop the display that the user will be seeing and sending it back to the (MM).

6.3 module-3 GMap

6.3.1 **-** The GMap Module (GMM) is responsible Display a MapView. It is to receive the data from the (MM) and send a map visual correspond from data requested back to the (MM).

6.4 module-4 Traffic Request

6.4.1 **-** The Traffic Request Module (TRM) is responsible for receiving the traffic request methods. Also (TRM) is responsible for importing and exporting a text file of the starting and ending nodes. (TRM) will tell the (MM) which algorithm to use.

6.5 module-5 Routing

6.5.1 **-** The Routing Module (RM) is responsible for receiving the Shortest Path Methods. The Routing Module will simulator which shortest path method to use when it determines the shortest path between the starting and destination node. The (RM) then sends back the info to the (MM).

6.6 module-6 Topology

6.6.1 **-** The Topology Module (TM) is responsible for generate a topology using the data from the (GMM). The (TM) will allow the user to export and import the topology in text format. Also (TM) will send the topology info to the after import (MM).

6.7 module-7 Simulator

6.7.1 **-** The Simulator Module(SM) will be responsible for receiving and processing data from (MM). (SM) stimulate a network using the resources, that (MM) collected from (TRM), (RM), (TM). (SRM) will stored the result of the simulation in CSV file locally.

7. Detailed Lower Level Component Design

7.1 The Main Package

7.1.1 Classification
This package contains all classes and xml files that are involved in the front-end design and controller.

7.1.2 Processing Narrative (PSPEC)
This package is an input output design. where user input template for the NSFNET design and output new template based on user input. In addition, it outputs data base on routing method.

7.1.3 Interface Description
JavaFX is used to generate an user interface with the addition of GMapsFX to implement Google Maps in the UI.

7.1.4 Processing Detail
It takes data from the results and processes the data into a graphical form.

7.1.4.1 Restrictions/Limitations
The user is restricted on editing an NSFNET topology only in location in the US

7.1.4.2 Performance Issues
There is a little latency issue on startup when initializing google map.

7.1.4.3 Design Constraints
A default template is fixed. User cannot change the style of UI

7.1.4.4 Processing Detail For Each Operation
The graph controller class is responsible for processing info into a graph. the GUI controller is responsible for the template for the UI

7.2 The Models Package

7.2.1 Classification
All classes are object to be called in other classes.

7.2.2 Processing Narrative (PSPEC)
Takes in parameters/arguments to generate an object

7.2.3 Processing Detail For Each Operation
User passes in arguments where the class has methods to help in the front end or back end

7.3 The Utilities Package

7.3.1 Classification

This contains Dijkstra's algorithm, the dynamic simulator add-on, and the topology.

7.4 Transponder Metric Class

7.4.1 Classification

This controls the simulator, where it passes the parameters in order to allow it to run.

7.5 Simulator Class

7.5.1 Classification

This class is responsible for generating the results based on the parameters passed thru the Transponder Metric class.

8. Database Design

We do not have any databases in our program.

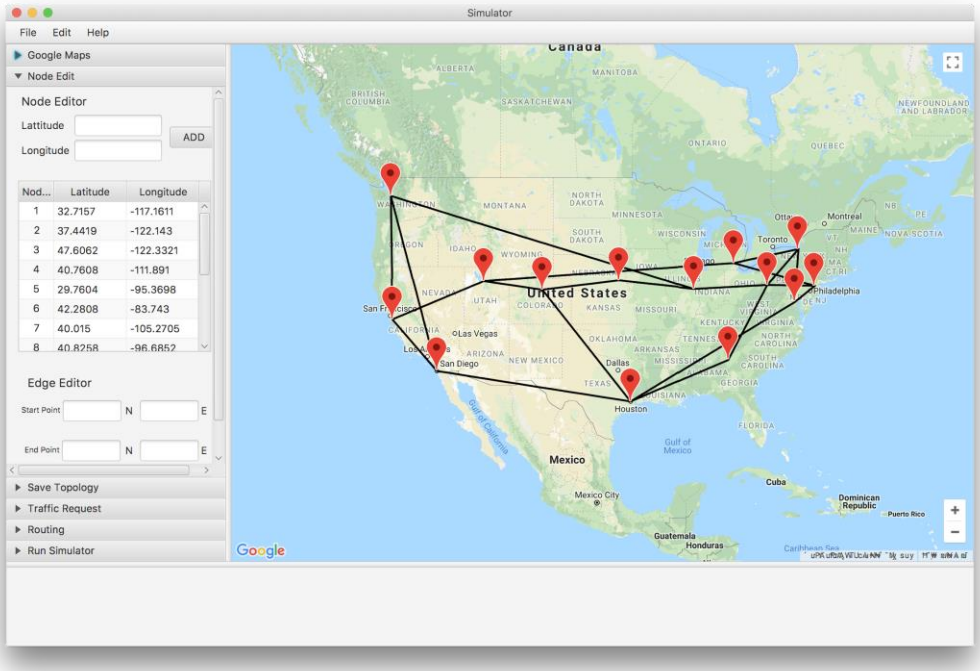
9. User Interface

9.1 Overview of User Interface

The user interface will contain Google Maps to right side and on the left side a series of tabs. Each tab contains functions that allows the user to interact with the simulator and Google Maps

9.2 Screen Frameworks or Images

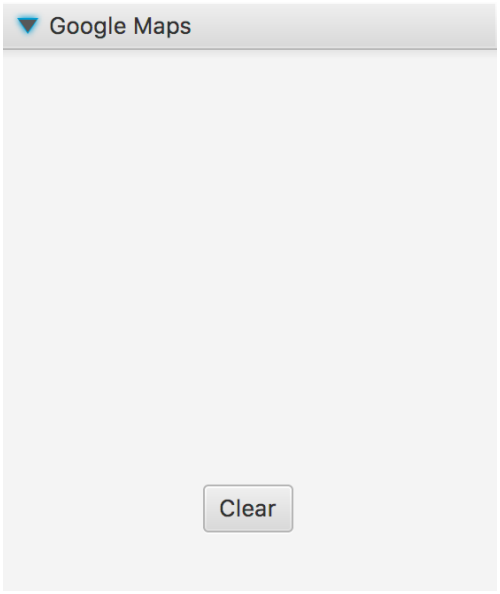
These can be mockups or actual screenshots of the various UI screens and popups.



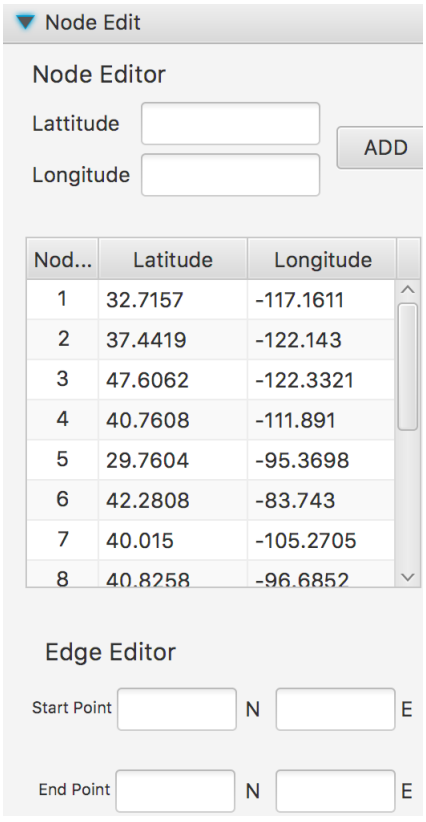
9.3 User Interface Flow Model

A discussion of screen objects and actions associated with those objects. This should include a flow diagram of the navigation between different pages.

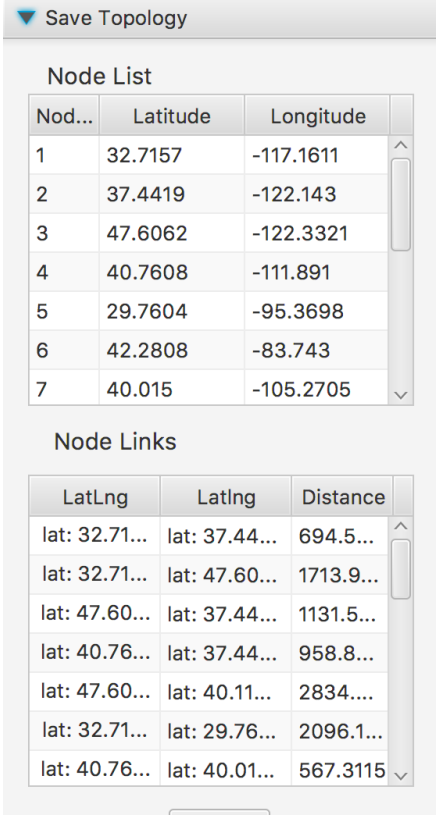
On the main page, we have a side tab with options on the left and a Google map on the right. For the options, we have Google Maps, Node Edit, Save Topology, Traffic Request, Routing, and Run Simulator.



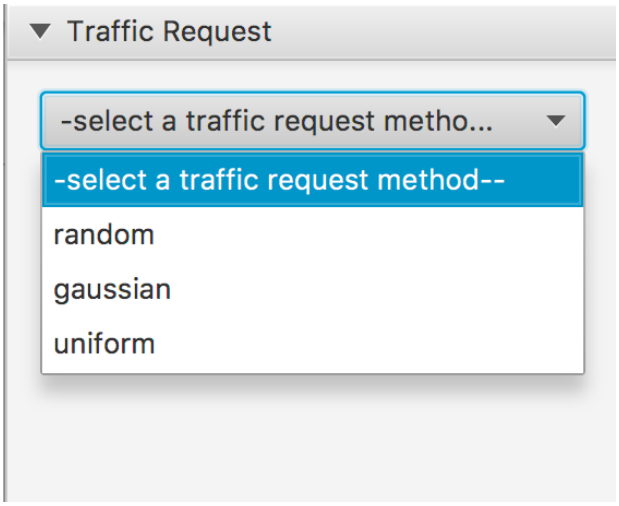
Google Maps enables the user to clear all nodes on the map.



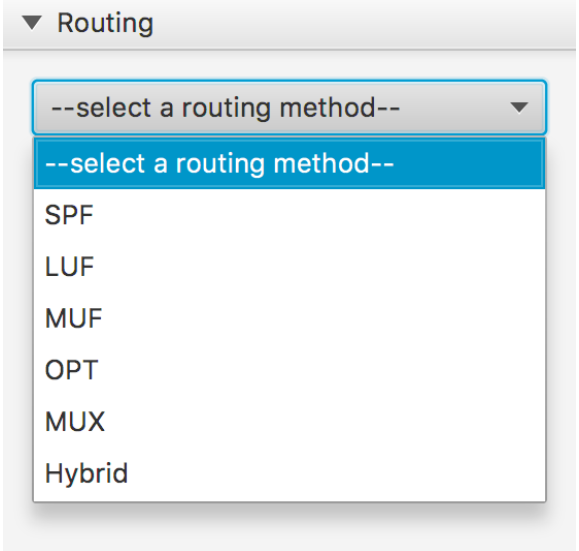
Node Edit enables the user to add nodes and edges.



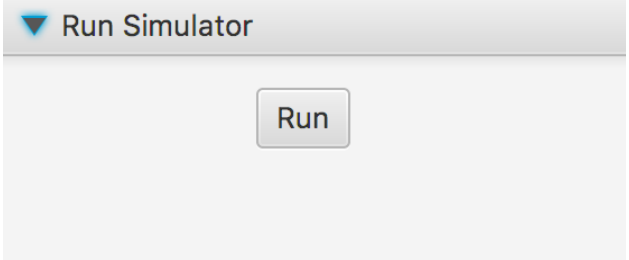
Save Topology enables the user to save the current topology and links to the topology file.



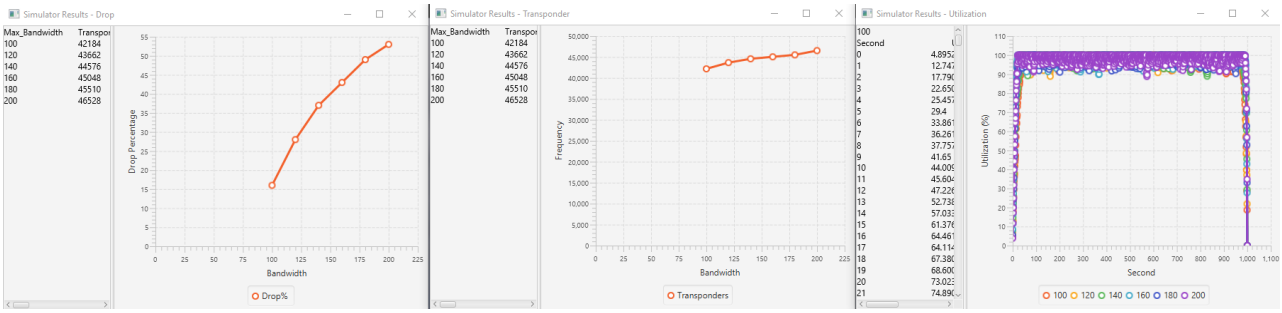
Traffic Request enables the user to choose the traffic request method to run for the simulator.



Routing enables the user to choose the routing method to run for the simulator.



Run Simulator enables the user to run the simulator with the selected settings.



After the simulator is run, there will be pop-ups with the result graphs shown.

10. Requirements Validation and Verification

Requirements Related to Design Module 4.1 **GUI Module**		Method for Testing
Requirement No.	Requirement Description	
10.1.1	The GUI shall have a view of the map.	QA
10.1.2	The GUI shall have a Node Edit section.	QA
10.1.3	The GUI shall have a function that will allow the user to disable a node.	QA
10.1.4	The GUI shall have a Topology section.	QA
10.1.5	The GUI shall have a function to save and import a topology.	QA
10.1.6	The GUI shall have a Traffic Request section.	QA
10.1.7	The GUI shall have a drop down menu that will allow the user to select between various traffic request methods.	QA
10.1.8	The GUI shall have a function that will allow the user to select between randomly generating requests or importing his own set of requests.	QA
10.1.9	The GUI shall have a Routing section.	QA
10.1.10	The GUI shall have a function that will allow the user to select between various routing methods.	QA
10.1.11	The GUI shall have a Simulator section	QA
10.1.12	The GUI shall have a run button that will start the simulator.	QA

Requirements Related to Design Module 4.2 **GMap Module**		Method for Testing
Requirement No.	Requirement Description	
10.2.1	GMap module shall allow user to select points or location on Google Maps corresponding to the longitude and latitude.	QA
10.2.2	GMap module shall allow user to draw diagram in between the two or multiple nodes.	QA
10.2.3	GMap module shall allow user to delete nodes.	QA
10.2.4	GMap module shall allow user to enter latitude and longitude manually.	QA
10.2.5	GMap module shall display all the entries of nodes.	QA

Requirements Related to Design Module 4.3 **Traffic Request(TR)**		Method for Testing
Requirement No.	Requirement Description	
10.3.1	TR module shall allow user to select different traffic method.	QA
10.3.2	TR module shall display three traffic method to user by default such as random, gaussian and uniform.	QA

10.3.3	TR module shall interact with simulator which randomization algorithm to use.	QA
10.3.4	TR module shall allow user to import custom requests.	QA

Requirements Related to Design Module 4.4 **Routing Module(RM)**		Method for Testing
Requirement No.	Requirement Description	
10.4.1	RM module shall allow user to choose between different routing algorithms.	QA
10.4.2	RM module shall display six different routing algorithm by default such as SPF,LUF,MUF,MUX,OPT and Hybrid.	QA
10.4.3	RM module shall display an option to use the dynamic simulator if SPF, LUF, or MUF is chosen.	QA
10.4.4	RM module shall interact with simulator module which shortest path to use upon user selection.	QA

Requirements Related to Design Module 4.5 **Topology Module(TM)**		Method for Testing
Requirement No.	Requirement Description	
10.5.1	TM module shall retrieve the data from the GMap module	QA
10.5.2	TM module shall save all the data into .txt format and store it locally	QA

Requirements Related to Design Module 4.6 **Simulator Module(SM)**		Method for Testing
Requirement No.	Requirement Description	
10.6.1	SM module shall interact with TR module which request method been selected.	QA
10.6.2	SM module shall interact with RM module which shortest path algorithm been selected.	QA
10.6.3	SM module shall generate the bandwidth process and display it into 2D graph format.	QA
10.6.4	SM module shall output the data into .csv format and store it locally.	QA

11. Glossary

SPF	This stands for Shortest Path First. This is a routing algorithm that determines the number of transponders based on the shortest path between the starting and destination node.
KSP	This stands for K-Shortest Path. This is a routing algorithm that determines K number

	of shortest paths between the starting and destination node.
LUF	This stands for Least Used First. This is a routing algorithm that first uses KSP to determine a number of shortest paths between the starting and destination node and then picks a path that has the least used transponder.
MUF	This stands for Most Used First. This is a routing algorithm that works the same way as LUF; however, this algorithm picks the path that has the most used transponder.
OPT	This is a routing algorithm that determines the number of transponders based on the direct link between the starting and destination node.
MUX	This is a routing algorithm is similar to OPT; however, this tries to minimize the number of transponders used by combining requests that have the same starting and destination node.
Hybrid	This is a routing algorithm that uses a threshold system where, if the traffic bandwidth is over a set threshold, OPT is run. Otherwise, SPF will run.
QA	This stands for quality assurance.

12. References

GMapsFX

http://rterp.github.io/GMapsFX/

Gluon Scene Builder

http://gluonhq.com/products/scene-builder/