Software Requirements Specification

for

Network Simulator

Version 2.0 approved

Prepared by:
Andy Do
William Fong
Dibakar Barua
Daniel Romo
Zifan Yang
Y Hoang

04/14/18

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
First Draft	11/27/17	Initial work on document.	0.1
Initial Submission	12/8/17	Finalized all sections	1.0
Final Submission	4/14/18	Added final requirements	2.0

1. Introduction

This is the general overview of the Network Simulator.

1.1 Purpose

This document will define all of the software requirements for the Network Simulator, also known as NS from this point forward. This will cover all of the modules and their intended purposes. All the modules described will make up NS as a whole. How the modules communicate with each other will be described in this document.

1.2 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. 2.3 - Product Functions
2. 2.4 - Operating Environment
3. 2.5 - Design and Implementation Constraints
4. 3 - External Interface Requirements

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

1.3 Product Scope

The proposed Network Simulator *NS* will be developed to show the effect of data transfer over a given network topology. This application will be able to generate streams of requests for the topology that are given by the user and output the logged data to a visualizer, where its interfacing is done through a file.

Successful implementation of the application will help network administrators to build network diagrams. In addition, the administrators will be able to run simulations to compare various traffic routing algorithms.

1.4 Definitions, Acronyms, and Abbreviations

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.

1.5 References

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

2. Overall Description

This section provides the general descriptions about the NS.

NS is a network simulating application that simulates different network topologies. The main focus is the NSFNET topology where the topology is displayed in a Google Map view. The Google Map view is interactive, letting the user add new nodes into the map which adds a new node into the NSFNET topology. NS will also let users select different given traffic requests and routing methods which uses the random algorithm and the routing method of user choice. In the end of each simulation, a table will appear displaying the results and will be stored locally.

2.1 Product Perspective

This application relies on Google Maps to display a map on the main screen. Having Google Maps implemented allows us to select various points to map out a topology, which are used for our simulation portion of the program. Aside from our use of Google Maps, this is a standalone application.

2.2 Product Functions

This product will have one major function that is to allow the user to run a simulator to determine the best traffic routing algorithm with a given topology. The modules that support this function are described below:

- GUI
    - This module will display the application Graphical User Interface (GUI) on the user's display. The user will navigate through this application with mouse and keyboard inputs. All input from the mouse and keyboard will be processed through the GUI, which will then send the information to the respective module that requests it.
- GMap
    - The Google Maps (GMap) module will send map data to the map view in the software. It will allow the user to add and delete nodes from a node list. It shall allow the user to click on any point on a map to add the coordinates of the point to the node list. In addition, the user will be able to disable a node to simulate an earthquake, where the affected node is suffering an outage.
- Traffic Request
    - This module shall allow the user to select between three different traffic request methods: random, gaussian, and uniform. This will tell the simulator which randomization algorithm to use when it generates a bandwidth used for each request. In addition, the user will be able to choose between letting the simulator randomly generate a random set of requests or use a custom set of requests. The user will be able to import a text file containing sets of starting and destination nodes if he decides to use a custom set of requests.
- Routing

    - This module shall allow the user to select between six different shortest path methods: SPF, LUF, MUF, OPT, MUX, and Hybrid. This will tell the simulator which shortest path method to use when it determines the shortest path between the starting and destination node.
- Topology
    - This module shall allow the user to generate a topology using the data from the GMap module. The user will be able to save the topology into a text file that is stored locally. Alternatively, the user will be able to import his own topology from a text file
- Simulator
    - This module shall take the traffic request method, routing method, topology, and requests from their respective modules and run a simulator based on them. It will output its results into a .csv file that is stored locally.

2.3 User Classes and Characteristics

The NSA will be used by various users. The regular user will be able to use the NSA without any difficulty. Our target audience are those who are technical and interested in learning about networking algorithms.

2.4 Operating Environment

The NSA is built using Java as its base. JavaFX will be used for our GUI. The NS will work on any platform, provided that the user has the latest Java Runtime Environment installed.

2.5 Design and Implementation Constraints

- The network topology is simple and static, which is represented by a simple undirected graph.
- The links are full duplex.
- Simulation of errors including packet losses, retransmission and noise burst are handled analytically.
- Routing in the network are fixed.

2.6 User Documentation

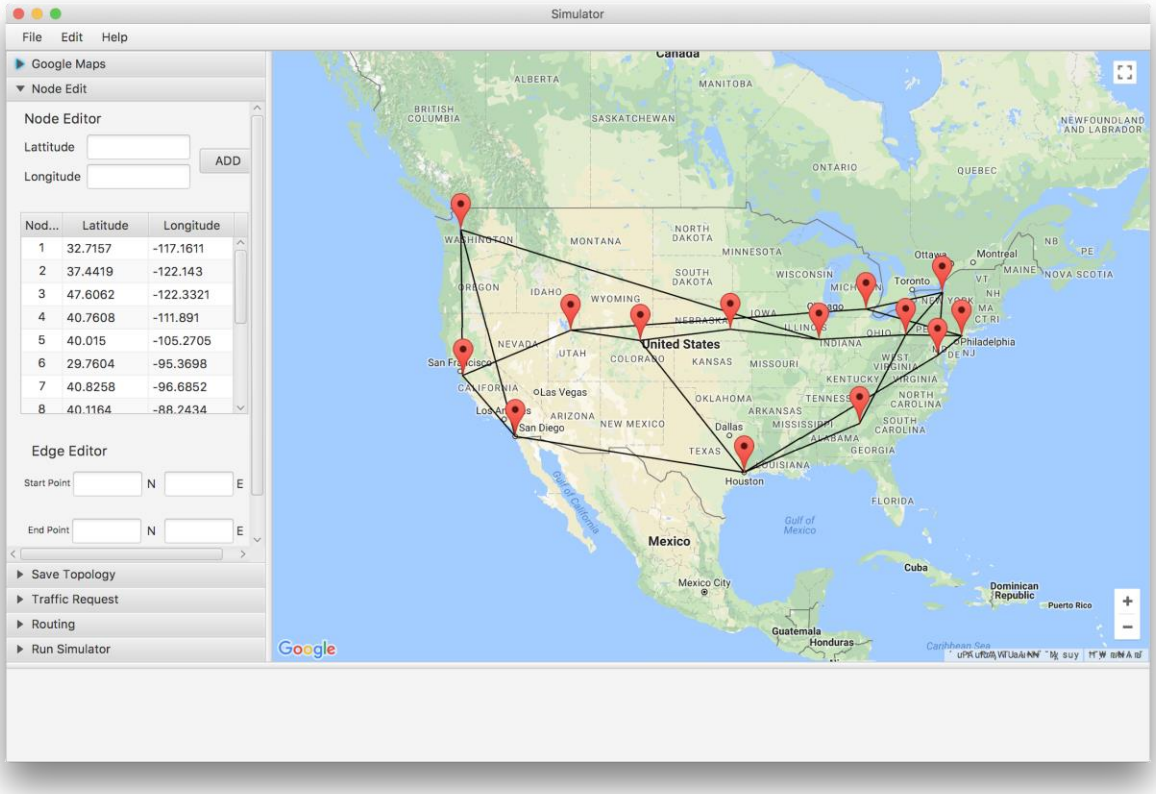
No user documentation will be required for this application.

2.7 Assumptions and Dependencies

A dependency for the NSA is GMapsFX, linked here: http://rterp.github.io/GMapsFX/.

3. External Interface Requirements

3.1 User Interfaces



This is the main interface of the software. The options are on the left sidebar, and the Google Maps is shown on the right side.

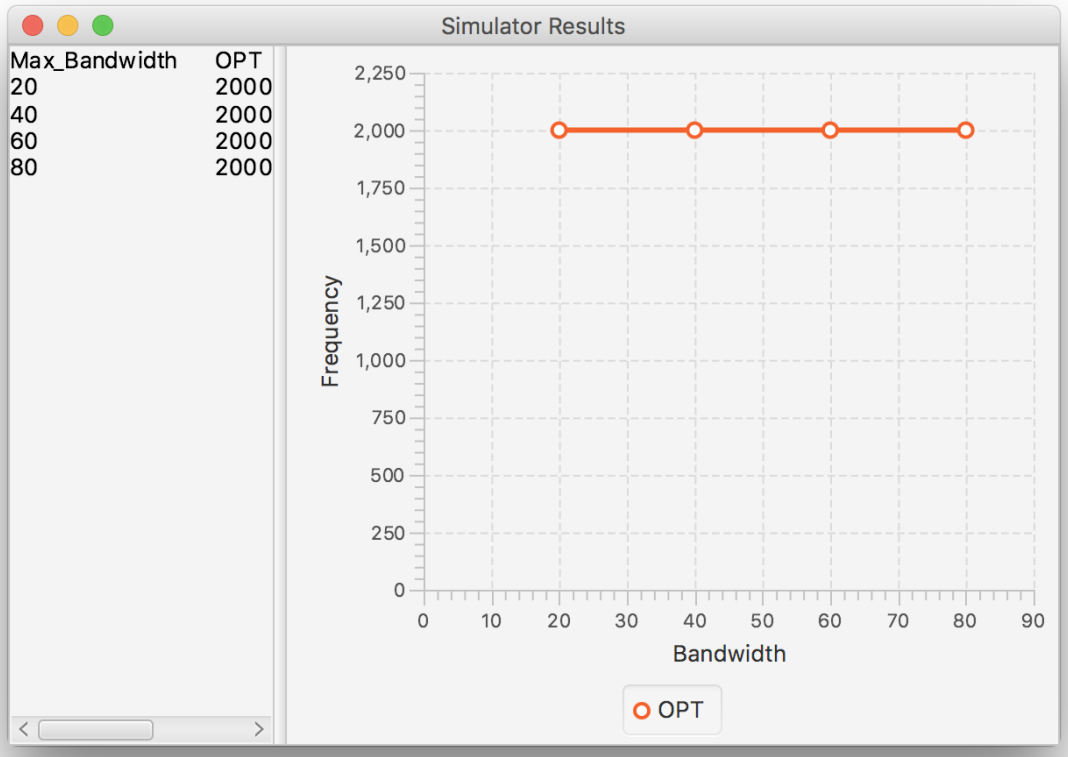
In the Node Edit tab, the user can add a node to the map by specifying the specific latitude and longitude with the Node Editor. It also shows a list of all the current nodes on the map. The Edge Editor allows the user to specify two nodes and connect them together. The user can either enter the node coordinates manually or by clicking on the pin on the map.

In the Save Topology tab, it allows the user to see all the links between the nodes.

In the Traffic Request tab, the user specify which type of traffic request you want to run in the simulator.

In the Routing tab, the user chooses which routing algorithm you want to run.

In the Run Simulator, the user runs the simulator with the specified settings on the previous tabs.



A window with the results will pop up showing a chart and graph of the simulation run.

3.2 Hardware Interfaces

Our software requires a computer that can run Java applications and has access to the Internet. It will require the use of a mouse for input and displays the results with a monitor.

3.3 Software Interfaces

The NS will require Java to run the user interface. It uses JavaFX for the front end. The front end was developed in FXML using SceneBuilder. We utilize the GMapsFX API to read and write points onto the map.

3.4 Communications Interfaces

The software will communicate with Google Maps servers using the GMapsFX API to generate a map and display additional information onto the map through the requests. All other requests are pulled locally.

4. Requirements Specification

4.1 Functional Requirements

This Section collects all **Network Simulator or NS** Functional Requirements. The Section includes the complete set of functional requirements with explanation and rational where the statement of the requirement was deemed insufficient or needing additional background/justification. All requirements relate to the design modules described in Section 2. An effort has been made to standardize the correlation between the design modules and the requirements to make their access and organization more consistent.

Requirements Related to Design Module 4.1 **GUI Module**	
Requirement No.	Requirement Description
4.1.1	The GUI shall have a view of the map.
4.1.2	The GUI shall have a Node Edit section.
4.1.3	The GUI shall have a function that will allow the user to disable a node.
4.1.4	The GUI shall have a Topology section.
4.1.5	The GUI shall have a function to save and import a topology.
4.1.6	The GUI shall have a Traffic Request section.
4.1.7	The GUI shall have a drop down menu that will allow the user to select between various traffic request methods.
4.1.8	The GUI shall have a function that will allow the user to select between randomly generating requests or importing his own set of requests.
4.1.9	The GUI shall have a Routing section.
4.1.10	The GUI shall have a function that will allow the user to select between various routing methods.
4.1.11	The GUI shall have a Simulator section
4.1.12	The GUI shall have a run button that will start the simulator.

Requirements Related to Design Module 4.2 **GMap Module**	
Requirement No.	Requirement Description
4.2.1	GMap module shall allow user to select points or location on Google Maps corresponding to the longitude and latitude.
4.2.2	GMap module shall allow user to draw diagram in between the two or multiple nodes.
4.2.3	GMap module shall allow user to delete nodes.
4.2.4	GMap module shall allow user to enter latitude and longitude manually.
4.2.5	GMap module shall display all the entries of nodes.

Requirements Related to Design Module 4.3 **Traffic Request(TR)**	
Requirement No.	Requirement Description

4.3.1	TR module shall allow user to select different traffic method.
4.3.2	TR module shall display three traffic method to user by default such as random, gaussian and uniform.
4.3.3	TR module shall interact with simulator which randomization algorithm to use.
4.3.4	TR module shall allow user to import custom requests.

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

Requirements Related to Design Module 4.5 **Topology Module(TM)**	
Requirement No.	Requirement Description
4.5.1	TM module shall retrieve the data from the GMap module
4.5.2	TM module shall save all the data into .txt format and store it locally

Requirements Related to Design Module 4.6 **Simulator Module(SM)**	
Requirement No.	Requirement Description
4.6.1	SM module shall interact with TR module which request method been selected.
4.6.2	SM module shall interact with RM module which shortest path algorithm been selected.
4.6.3	SM module shall generate the bandwidth process and display it into 2D graph format.
4.6.4	SM module shall output the data into .csv format and store it locally.

4.2 External Interface Requirements

This section will go through the input and output of each module.

Main Controller

- The main controller will handle communication with all the other modules. It will send data received from other modules to the appropriate module.

GUI

- The GUI will be what the user sees upon starting the application.
- Data input from the user will come from actions such as mouse clicks and keyboard entry.

GMaps

- This module will handle input from Google Maps.
- This module will handle creation, editing, and disabling of nodes.

Traffic Request

- This module will allow the user to choose between several traffic request methods.
- This module will allow the user to import his own set of requests or have the simulator generate its own raser ndom set.

Topology

- This module consist of three randomization method by default such as gaussian, randomization and uniform. This module will also allow user to import custom topology according to their needs.

Simulator

- This module will generate the bandwidth process and show the output in 2D format in terms of the traffic request and topology method that are selected by the users.

4.3 Logical Database Requirements

The Network Simulator NS software will store data into both .txt and .csv file format. Listed below are the attributes that will be stored locally into .txt and .csv file.

- Latitude
- Longitude
- Distance
- Bandwidth

4.4 Design Constraints

There are no design constraints for this software.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The application will meet and follow the following performance requirements:

- Unlimited amount of Nodes can be created on the map.

- Unlimited amount of Edges can be created.
- The program will export into the CSV file type.
- The program will take a CSV type file from the import function.

5.2 Safety Requirements

There are no safety requirements.

5.3 Security Requirements

The application generally can be operate offline, aside from the map module is the only module require an internet connect to render the map. The Google Map API requires a secure connection.

5.4 Software Quality Attributes

The NS is made to be used for Windows or Mac with various screen resolution sizes. It is responsive to window size changes with its elegant implementation. Functions will have descriptions displayed when hovered over.

5.5 Business Rules

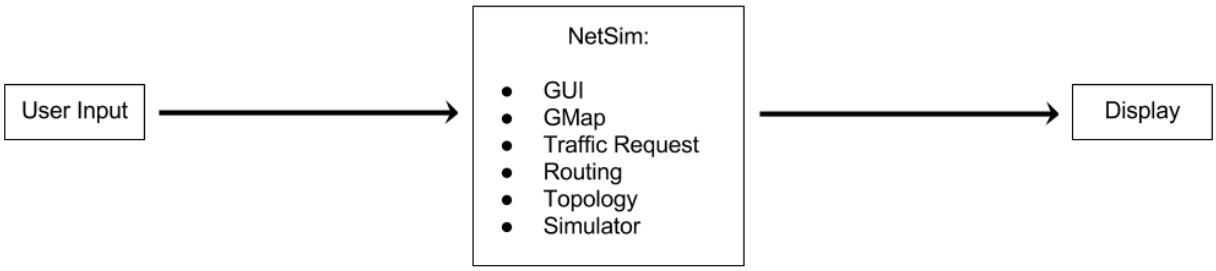
Any user will be able to use the NS.

6. Other Requirements

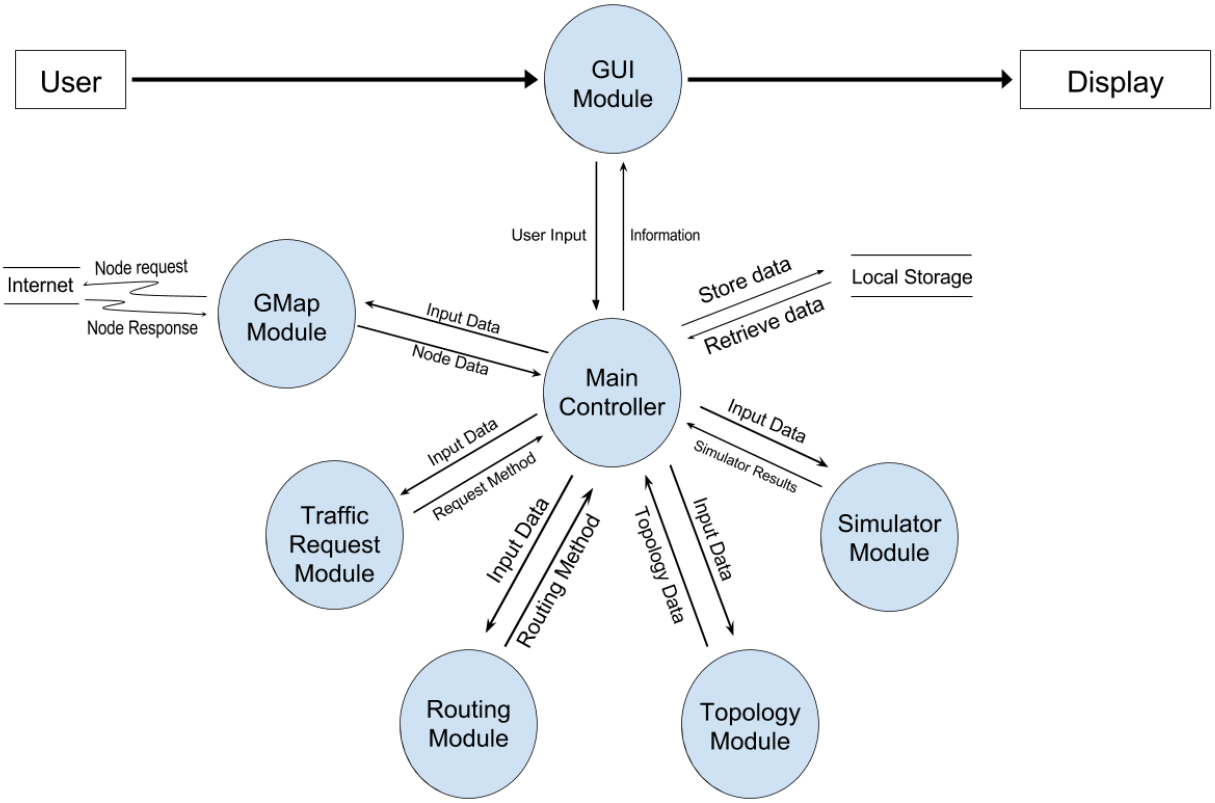
Appendix A: Glossary

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.

Appendix B: Analysis Models



Level 0 DFD



Level 1 DFD