NetAnalyzer: Statistics for Dynamic & Static Networks

User Manual

Supervising Professor: Thomas Kunz

An ongoing project with contributions by Evan Bottomley, Cristopher Briglio, Henri Cheung, Jordan Edgecombe and Hussain Saeed

Version 4

Carleton University

Summer 2015

Table of Contents

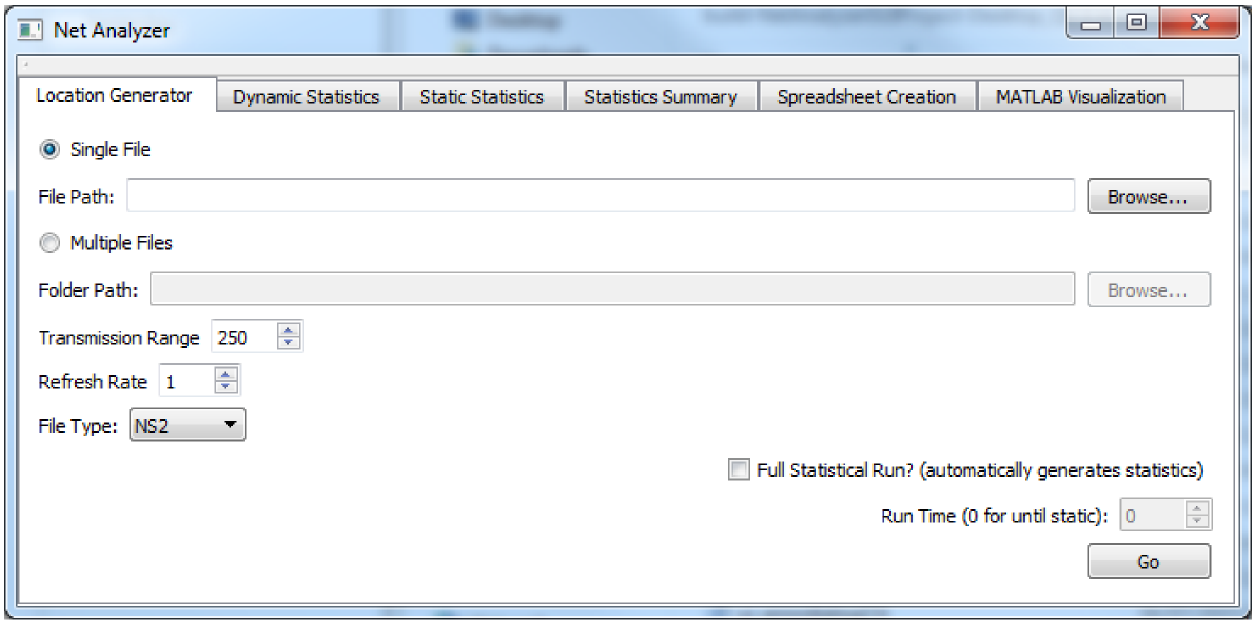
About

NetAnalyzer: Statistics for Dynamic & Static Networks is a program written and compiled in C++. The program creates statistics from dynamic and static network simulations of NS2, Opnet and Core formats using the *Boost Graph Library*. The program also uses MATLAB, GraphViz, and Excel to help visualize, organize and interpret the statistics. Finally, Qt Creator is used to create a GUI.

Title Screen

The main window is contained in *mainwindow.ui, mainwindow.h* and *mainwindow.cpp*. Launching the GUI will default to the Location Generator tab.

Location Generator



(●) Single File

File Path:

( ) Multiple Files

Folder Path:

Transmission Range 250

Refresh Rate 1

File Type: NS2

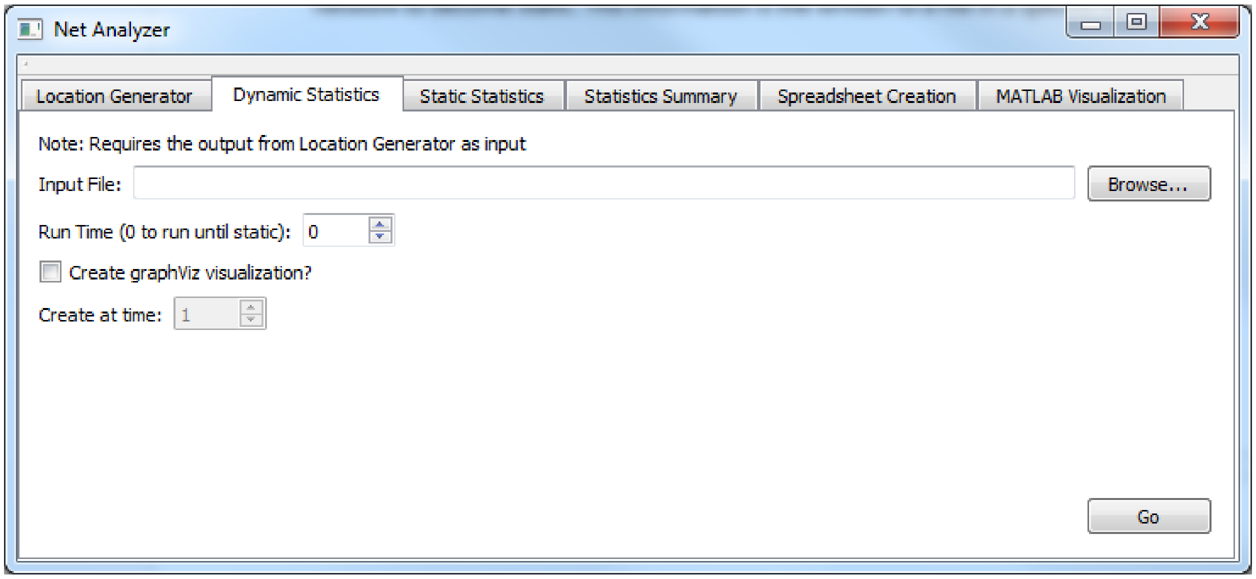
☐ Full Statistical Run? (automatically generates statistics)

Run Time (0 for until static): 0

Location Generator can interpret NS2, Opnet and Core dynamic network simulations. It gives the user the choice to generate an output file for single or multiple files at a time. In the case of multiple files, the user must specify a folder location for the program to look through. After locating the file(s) specified, the program will create an output folder for the scenario(s). The name of the folder has the format: '**scenario file name** output'. The program then reads in the scenario and generates each nodes location every second until the node becomes static, retaining the longest run time needed for the network to become static. This information is the written to a file in a specific format. If "Full Statistical Run" is checked, then the program will run Dynamic Statistics right after the output file is generated.

Output for the Location Generator is located in a folder named *filename* output, in the same folder as the input file.

Dynamic Statistics



Note: Requires the output from Location Generator as input

Input File:

Run Time (0 to run until static): 0

☐ Create graphViz visualization?

Create at time: 1

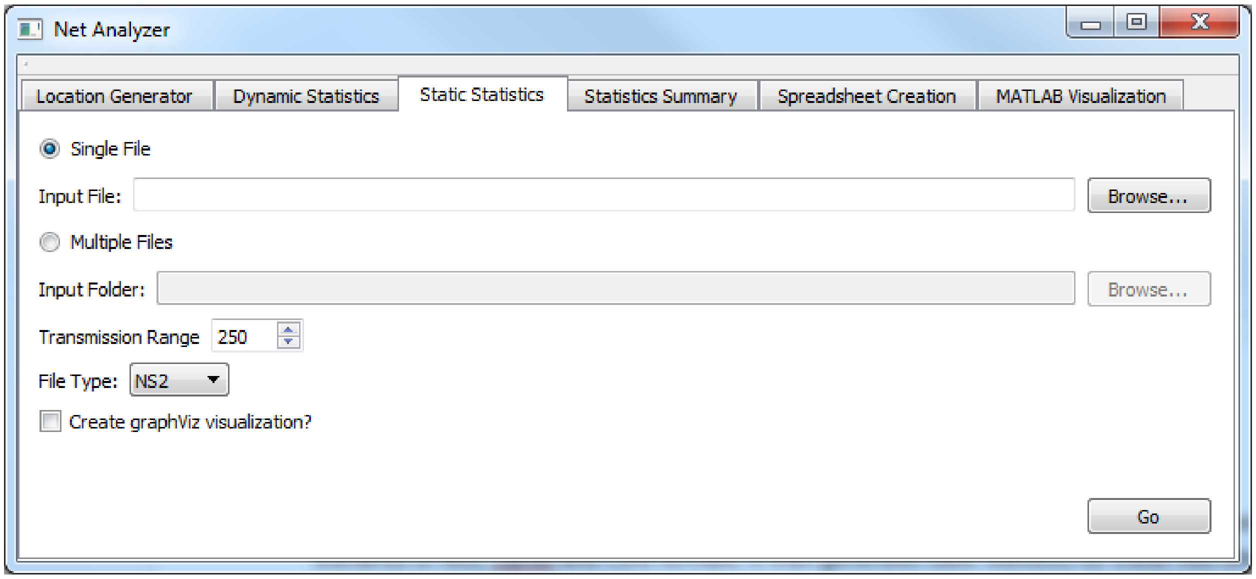
Dynamic Statistics reads the user specified output file from the *Location Generator* and creates statistics for the user chosen run time in intervals of the given refresh rate. Using the *Boost Graph Library* the following statistics are created:

- Average number of neighbors each node has
- The nodes that have the most neighbors
- The average distance between nodes
- The number of components
- The network diameter
- The nodes with the highest betweeness centrality
- The number of links changed
- The coverage of the combined nodes over time
- The network instability
- The clustering coefficients
- The degree of randomness/predictability

The statistics are then written to their own separate files within the output folder. It also gives the option to create a graphViz visualization of the network at any given multiple of the refresh rate.

Output files are located in the same folder as the input for Dynamic Statistics

Static Statistics


Net Analyzer

Location Generator | Dynamic Statistics | Static Statistics | Statistics Summary | Spreadsheet Creation | MATLAB Visualization

[✓ Single File]

Input File: Browse...

Multiple Files

Input Folder: Browse...

Transmission Range: 250

File Type: NS2

Create graphViz visualization?

Go

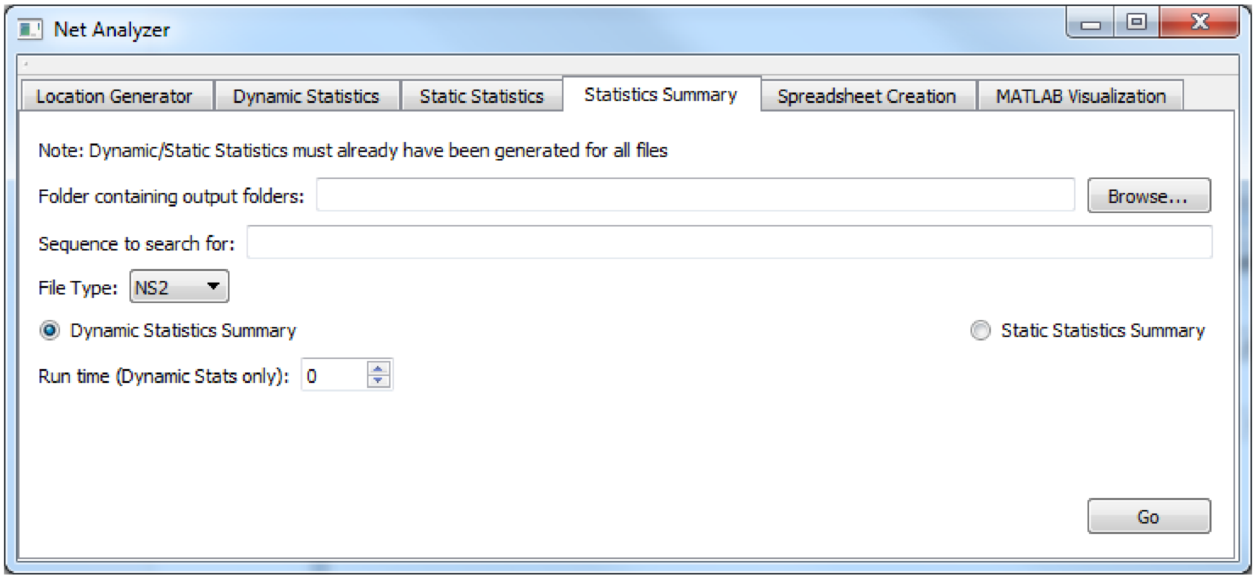
The Static Statistics portion of the program (originally written by Hassain Saeed) reads in static network scenarios of NS2, Opnet and Core formats. It then generates basic statistics for these files which include:

- Average number of neighbors each node has
- The nodes that have the most neighbors
- The average distance between nodes
- The number of components
- The network diameter
- The nodes with the highest betweeness centrality

The statistics are then written to their own separate files within an output folder. It also gives the option to create a graphViz visualization of the network.

Output is located in a folder called *filename* output, in the same folder as the input file.

Statistical Summary



Note: Dynamic/Static Statistics must already have been generated for all files

Folder containing output folders:

Sequence to search for:

File Type: NS2

(●) Dynamic Statistics Summary ( ) Static Statistics Summary

Run time (Dynamic Stats only): 0

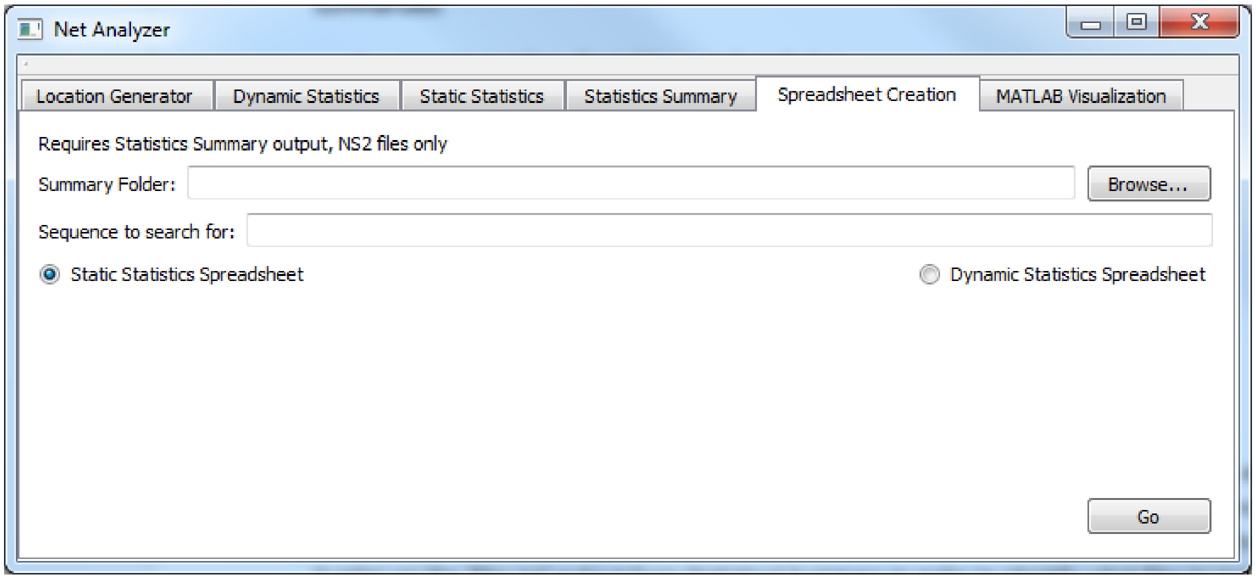
The Statistical Summary part of the program creates averaged statistics from multiple scenarios. The program works by reading the statistics generated from Dynamic Statistics or Static Statistics for several specific files and creates a summary of these statistics in a separate folder. The following statistics are summarised:

- average number of neighbors each node has
- the average distance between nodes
- the number of components
- the network diameter
- the number of links changed
- the percent coverage of the area
- the average neighbourhood instability
- the average clustering coefficient

(Links summary isn't included for static scenario summaries).

The output can be found in a folder called *sequence*.summary, located in the same folder as the input files

Spreadsheet Creator



Requires Statistics Summary output, NS2 files only

Summary Folder:

Sequence to search for:

(● Static Statistics Spreadsheet) ○ Dynamic Statistics Spreadsheet

The Spreadsheet Creator uses the statistics created in Dynamic Statistics or Static Statistics, as well as Statistical Summary part of the program to create a spreadsheet that can be opened in Microsoft Office Excel. The spread sheet brings together the specific data from all the scenarios in a visually friendly way. It relies on the *'files.txt'* output from Statistical Summary in order to identify what files to include. The spreadsheet includes the following data (for dynamic scenarios):

- Average links created and broken per second
- Total links created and broken in desired run time
- Number of connected components (at initial and final time)
- Average number of neighbours (at initial and final time)
- Average path length in hops (at initial and final time)
- Network diameter in hops (at initial and final time)

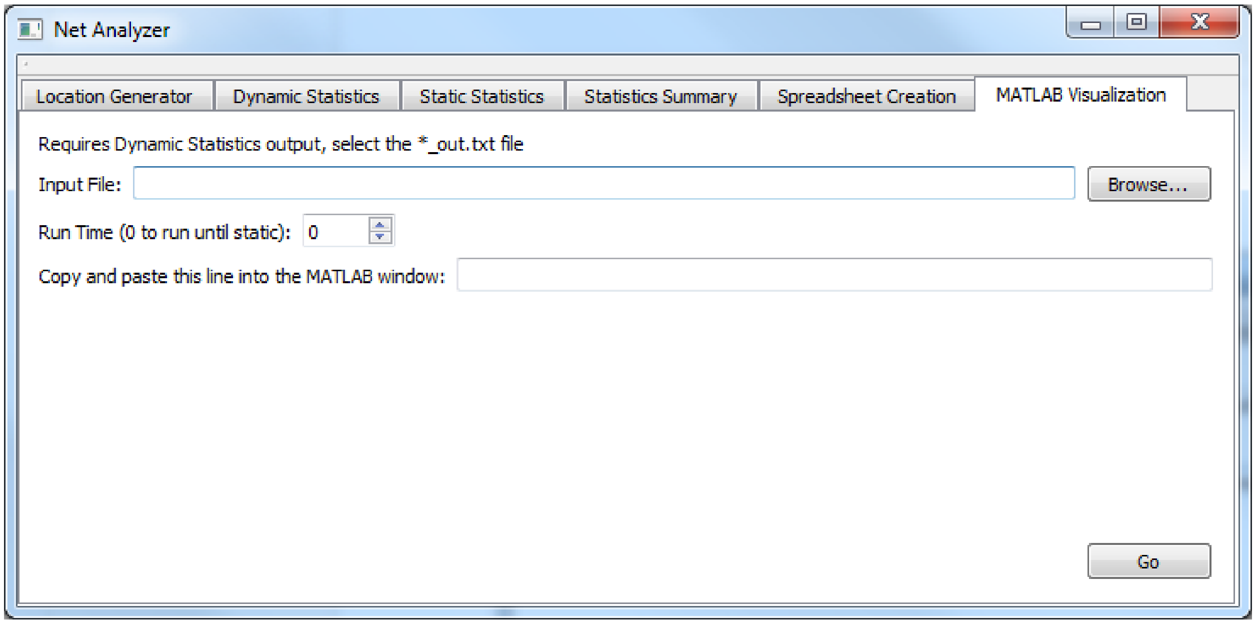
For static scenarios:

- Number of connected components
- Average number of neighbours
- Average path length in hops
- Network diameter in hops

This portion of the program only works on NS2 scenarios with file names with the format: 'Scenario#-#.txt'

Output is located in the *sequence*.summary folder.

MATLAB Visualization



Requires Dynamic Statistics output, select the *_out.txt file

Input File:

Run Time (0 to run until static): 0

Copy and paste this line into the MATLAB window:

Animation Graphs is a program created in MATLAB that uses the statistics created in Dynamic Statistics for a particular scenario to create an animation of the nodes and their trajectories over time on an x-y plain for the user desired run time as well as several graphs illustrating all the statistics. Note that only the average clustering coefficient and neighbourhood instability are graphed, instead of the per-node results. Currently, this function does not have MATLAB automatically execute the script, so the user must paste the line indicated when the MATLAB window opens.

Building the Program

Required Software

Visual Studios 2012
    http://msdn.microsoft.com/library/dd831853.aspx
Boost Graph Library 1.54.0
    http://www.boost.org/doc/libs/1_54_0/libs/graph/doc/index.html
GraphViz 2.30.1
    http://www.graphviz.org/Download.php
MATLAB R2013a
    http://www.mathworks.com/help/matlab/index.html
Qt 5.4.2 (includes Qt Creator 3.4.1)
    http:/www.qt.io/download-open-source/

Microsoft Office Excel 2007
    http://office.microsoft.com/en-ca/excel/

If Building in Linux

OpenGL/GLUT 3
    http://kiwwito.com/installing-opengl-glut-libraries-in-ubuntu/
GCC 5.2
    https://help.ubuntu.com/community/InstallingCompilers

Setup

Building NetAnalyzer.exe

(At current stage of program is only compatible with windows, built using Microsoft Visual Studios 2012)

- In Visual Studio's *File* menu, select *New -> Project*. Then in the new window that pops up, under *Visual C++*, select *Win32 Console Application*.
- Delete the premade source and header files (right click -> exclude from project) and add all the necessary source and header files to the project. This includes:
    - *Main.cpp*
    - *Dynamic Statistics.cpp*
    - *Location Generator.cpp*
    - *Static Statistics.cpp*
    - *SpreadsheetCreator.cpp*
    - *stdafx.cpp*
    - *stdafx.h*
    - *convert.h*
    - *targetver.h*
- Apply all the required settings. The Property pages can be opened by selecting View -> Property Manager, then clicking the wrench symbol in the upper left corner.

Visual Studios Settings

The following are the setting that must be configures in *Visual Studious 2012* in order to run the program:

*Configuration Manager - Configuration - **Release***

Solution Configurations - ***Release***

Property Pages - Configuration Properties - General - Character Set - ***Not Set***

Property Pages - Configuration Properties - C/C++ - Precompiled Headers - ***Not Using***

(make sure Configuration is set to: *All Configurations*)

Add Boost library (boost_1_54_0 folder) to *Additional Included Directories* found in:

Property Pages - Configuration Properties - C/C++ - General - Additional Included Directories

- After this you simply build the project, which will create NetAnalyzer.exe in the "Release" folder.

Building NetAnalyzerGUI.exe

- In Qt creator, select *File -> New File or Project…* and select *Qt Widgets Application*.
- Choose a location and name, then make sure that the *Desktop Qt 5.4.2 MSVC2012 OpenGL 32bit* kit is selected (GCC instead of MSVC if building in Linux)
- Leave *Details* as the default names and finish starting the project.
- Copy the following files to the project folder, then include them
    - *Main.cpp*
    - *mainwindow.cpp*
    - *mainwindow.h*
    - *mainwindow.ui*
    - *errordialog.cpp*
    - *errordialog.h*
    - *errordialog.ui*
    - *errordialog2.cpp*
    - *errordialog2.h*
    - *errordialog2.ui*
- main.cpp, mainwindow.cpp, mainwindow.h and mainwindow.ui should already be included as they overwrite the default files.
- Build and run the Net Analyzer GUI through Qt Creator
- After building both NetAnalyzer and NetAnalyzerGUI, copy the NetAnalyzer.exe to the folder created when building NetAnalyzerGUI. Copy animations_graphs.m to the same folder. The folder should have a name similar to "build-NetAnalyzerGUIProject-Desktop_Qt_5_4_2_MSVC2012_OpenGL_32bit-Debug".
- Currently, the GUI runs primarily out of Qt Creator, however Qt Creator does build NetAnalyzerGUI.exe in the debug folder (which is in the folder from the previous step). To run NetAnalyzerGUI.exe, you must copy Qt5Cored.dll, Qt5Guid.dll and Qt5Widgetsd.dll to the debug folder. These can be found in the Qt folder, which is /path to Qt/Qt/5.4/msvc2012_opengl/bin. You must also copy NetAnalyzer.exe and animations_graphs.m to the debug folder.

Running The Program

The following is a run through of the program using these example files:

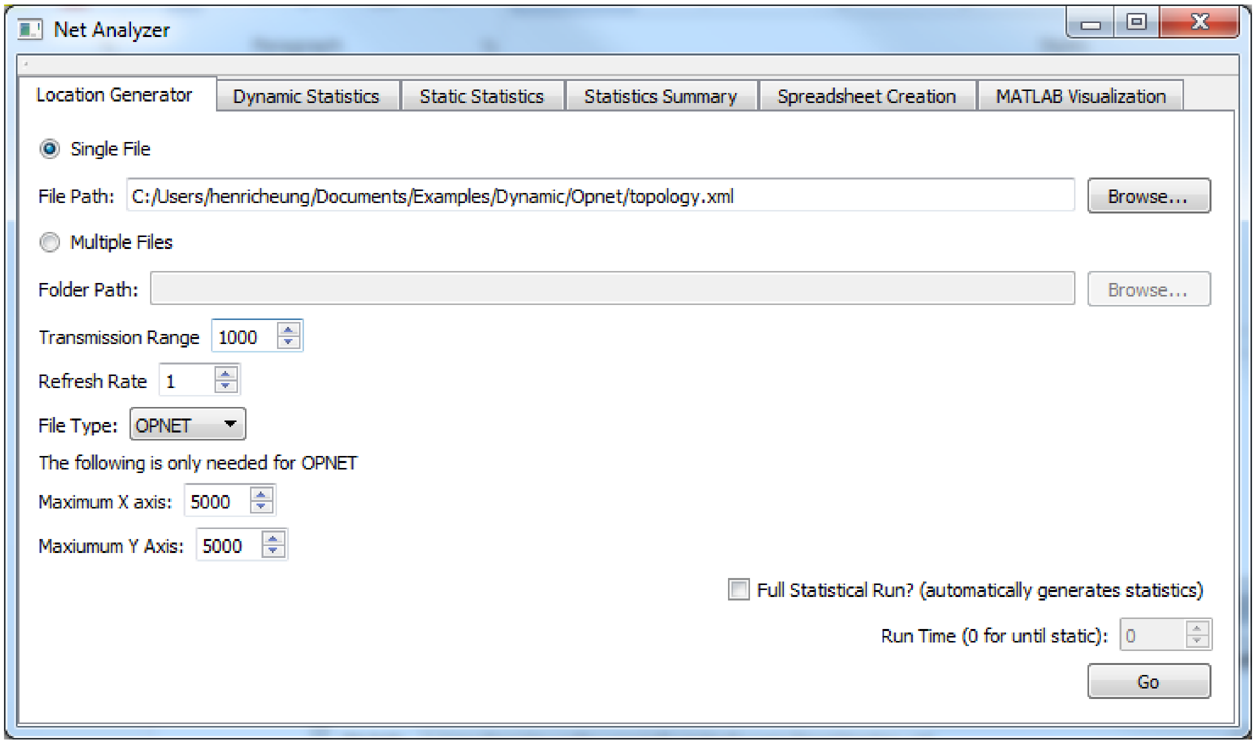
- Examples - Dynamic - Opnet - **topology.xml** (Single dynamic scenario)
- Examples - Dynamic - NS2 - **Dynamic** (Multiple scenarios)
- Examples - Static - NS2 - **Static** (Multiple scenarios)
- Examples – Dynamic – Core – **12node.imn** (Single dynamic scenario)

With the following examples the user should learn to run any aspect of the program.

(Different aspects of the program that are being shown may have been altered or changed since these pictures were taken as the program was further worked on. Also, these pictures may not depict the program running for these specific examples, please follow the written instructions to get the same results as the examples and not what the visuals show)

topology.xml (OPNET)

Location Generation



[● Single File]

File Path: C:/Users/henricheung/Documents/Examples/Dynamic/Opnet/topology.xml

Multiple Files

Folder Path:

Transmission Range 1000

Refresh Rate 1

File Type: OPNET

The following is only needed for OPNET

Maximum X axis: 5000

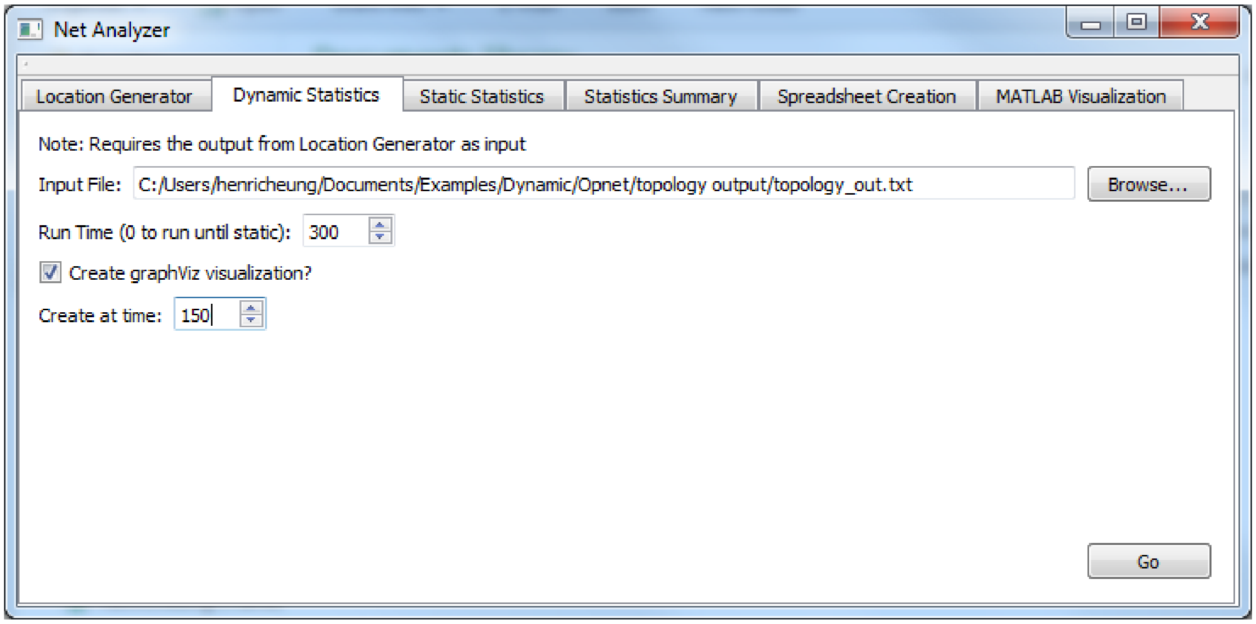
Maxiumum Y Axis: 5000

Full Statistical Run? (automatically generates statistics)

Run Time (0 for until static): 0

Here, the user has given the path to topology.xml, given a transmission range of 1000 and a refresh rate of 1 second. The maximum X and Y axis is also specified as 5000 for both. Also remember to specify the file type in the drop down box. Clicking "Go" will proceed with the operation and begin generating the output file: topology_out.txt.

Dynamic Statistics



Note: Requires the output from Location Generator as input

Input File: C:/Users/henricheung/Documents/Examples/Dynamic/Opnet/topology output/topology_out.txt

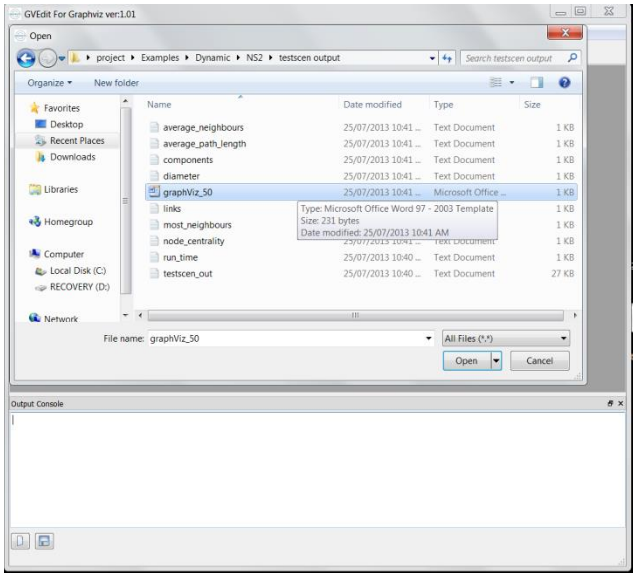
Run Time (0 to run until static): 300

☑ Create graphViz visualization?

Create at time: 150

Once the Location Generator is finished, the user should go to the Dynamic Statistics tab and select the output file, then choose a run time. In this case, the run time will be <u>300</u>, and a graphViz visual will be created at time <u>150</u>. Clicking "Go" will then proceed with collection of statistics using the given parameters.

GraphViz Visualization



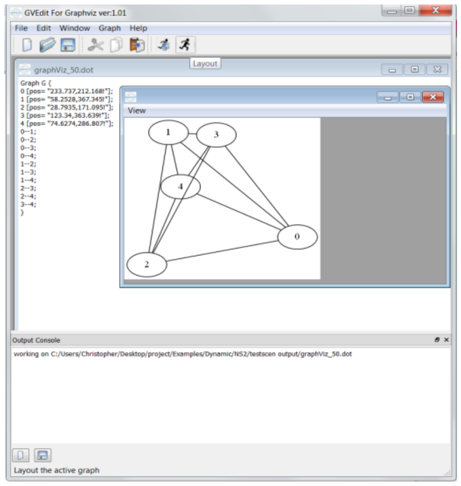
Name	Date modified	Type	Size
average_neighbours	25/07/2013 10:41 ...	Text Document	1 KB
average_path_length	25/07/2013 10:41 ...	Text Document	1 KB
components	25/07/2013 10:41 ...	Text Document	1 KB
diameter	25/07/2013 10:41 ...	Text Document	1 KB
graphViz_50	25/07/2013 10:41 ...	Microsoft Office ...	1 KB
links	[illegible]	[illegible]	1 KB
most_neighbours	[illegible]	[illegible]	1 KB
node_centrality	25/07/2013 10:41 ...	Text Document	1 KB
run_time	25/07/2013 10:40 ...	Text Document	1 KB
testscen_out	25/07/2013 10:40 ...	Text Document	27 KB

Type: Microsoft Office Word 97 - 2003 Template
Size: 231 bytes
Date modified: 25/07/2013 10:41 AM

File name: graphViz_50 All Files (*.*)

Once the GraphViz visualization has been created, the user must now use a separate program to create a photo of it. Within GraphViz, the *gvedit.exe* application can be used to read and output the visualizations. To start you must locate and open this application (found in: graphviz-2.30.1\release\bin\ gvedit.exe). Once running, have the program open the file by going to *file -> open*. Once the application has open the file (<u>graphViz_150.dot</u>) the user must then have it output the visualization as a picture. There are a few ways to do this although the easiest seems to be the following:

- Hit the <u>*Layout* button</u>, this button is located near the top left and has the shape of a man running. It can also be found within *Graph - > Layout.* Within the Output Console at the bottom of the screen it should say: *working on...*



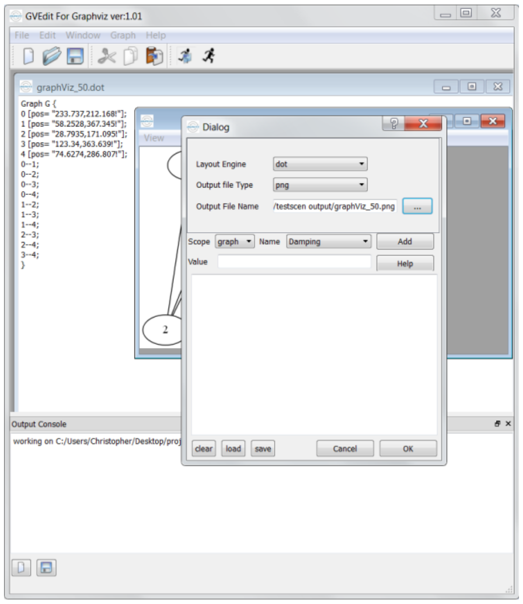
```
Graph G {
0 [pos= "233.737,212.168!"];
1 [pos= "58.2528,367.345!"];
2 [pos= "28.7935,171.095!"];
3 [pos= "123.34,363.639!"];
4 [pos= "74.6274,286.807!"];
0--1;
0--2;
0--3;
0--4;
1--2;
1--3;
1--4;
2--3;
2--4;
3--4;
}
```

Output Console

```
working on C:/Users/Christopher/Desktop/project/Examples/Dynamic/NS2/testscen output/graphViz_50.dot
```

Layout the active graph

- Next hit the *Settings* button located next to the layout button which looks like the layout button but with a blue box in it. It can also be found within *Graph -> Settings*.
- Once clicked it will bring up a menu name Dialog. Make sure its Layout Engine is *dot*, the Scope is *graph*, and specify your desired output file type then click *OK*



```
Graph G {
0 [pos= "233.737,212.168!"];
1 [pos= "58.2528,367.345!"];
2 [pos= "28.7935,171.095!"];
3 [pos= "123.34,363.639!"];
4 [pos= "74.6274,286.807!"];
0--1;
0--2;
0--3;
0--4;
1--2;
1--3;
1--4;
2--3;
2--4;
3--4;
}
```

Layout Engine: dot
Output file Type: png
Output File Name: /testscen output/graphViz_50.png
Scope: graph Name: Damping
Value:

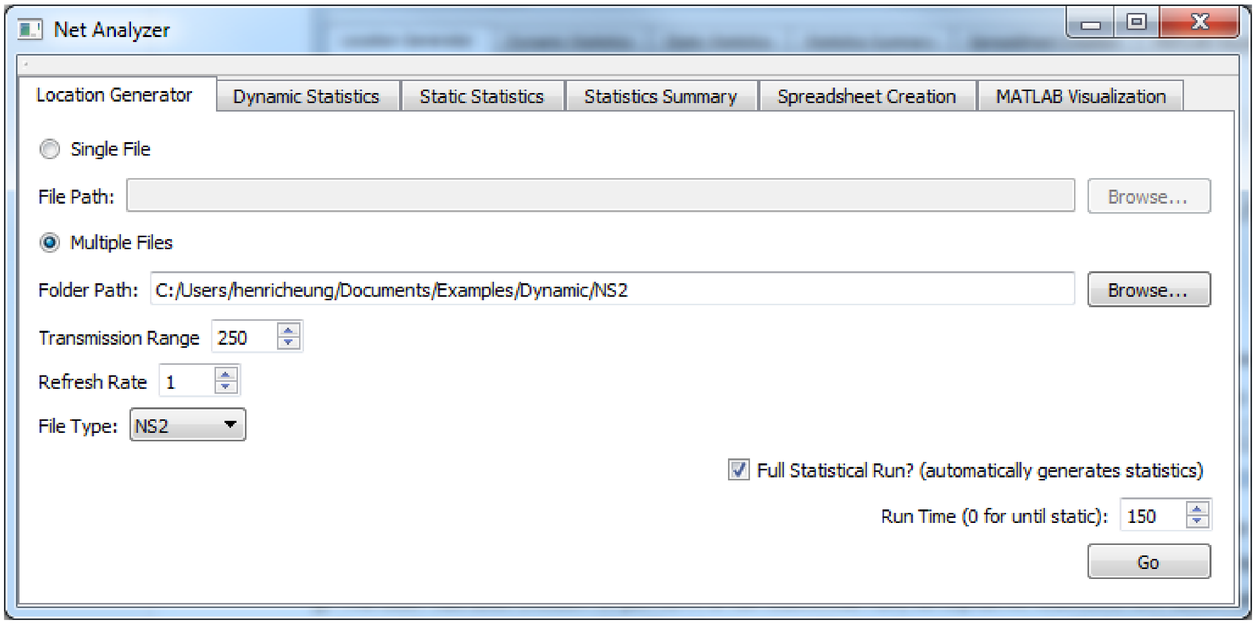
Output Console

```
working on C:/Users/Christopher/Desktop/proj
```

- Then click the Layout button two more times
- The visualization has now been output in the format that you specified

Dynamic Scenarios (NS2)

Full Statistical Run Through



Single File

File Path:

[✓ Multiple Files]

Folder Path: C:/Users/henricheung/Documents/Examples/Dynamic/NS2

Transmission Range 250

Refresh Rate 1

File Type: NS2

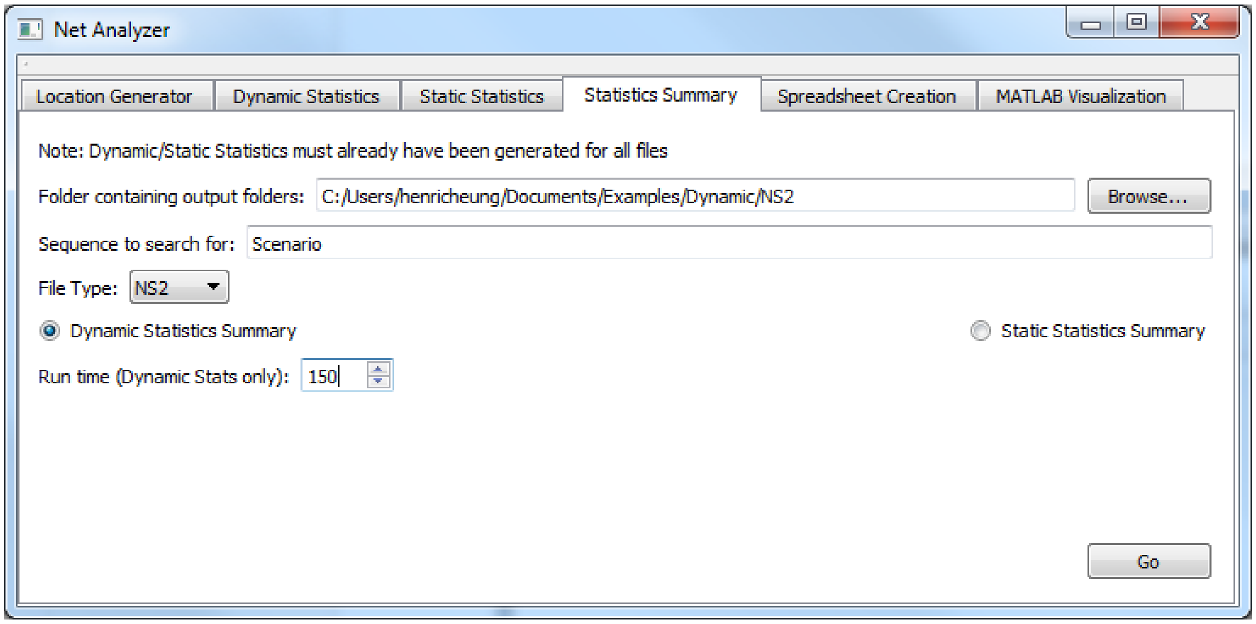
[✓] Full Statistical Run? (automatically generates statistics)

Run Time (0 for until static): 150

In this example, the user will be doing multiple file analysis with the Location Generator. The user gives the directory containing the scenario files, and a transmission range of 250 along with a refresh rate of 1. The user has also chosen to perform a full statistical run, so Dynamic Statistics will automatically be collected for every scenario file after their output file has been generated. Clicking "Go" will have the program begin analyzing all .txt files contained in the given directory (note: the program does not distinguish between scenario files and other .txt files, so it is recommended that there are only scenario files in the given directory). This is a VERY time-consuming process, considering the number of scenarios included in the Examples.zip, and can take half an hour or even more, depending on how powerful the user's machine is.

Statistics Summary

Next, the user should go to the Statistics Summary tab



Note: Dynamic/Static Statistics must already have been generated for all files

Folder containing output folders: C:/Users/henricheung/Documents/Examples/Dynamic/NS2

Sequence to search for: Scenario

File Type: NS2

[✓ Dynamic Statistics Summary] Static Statistics Summary

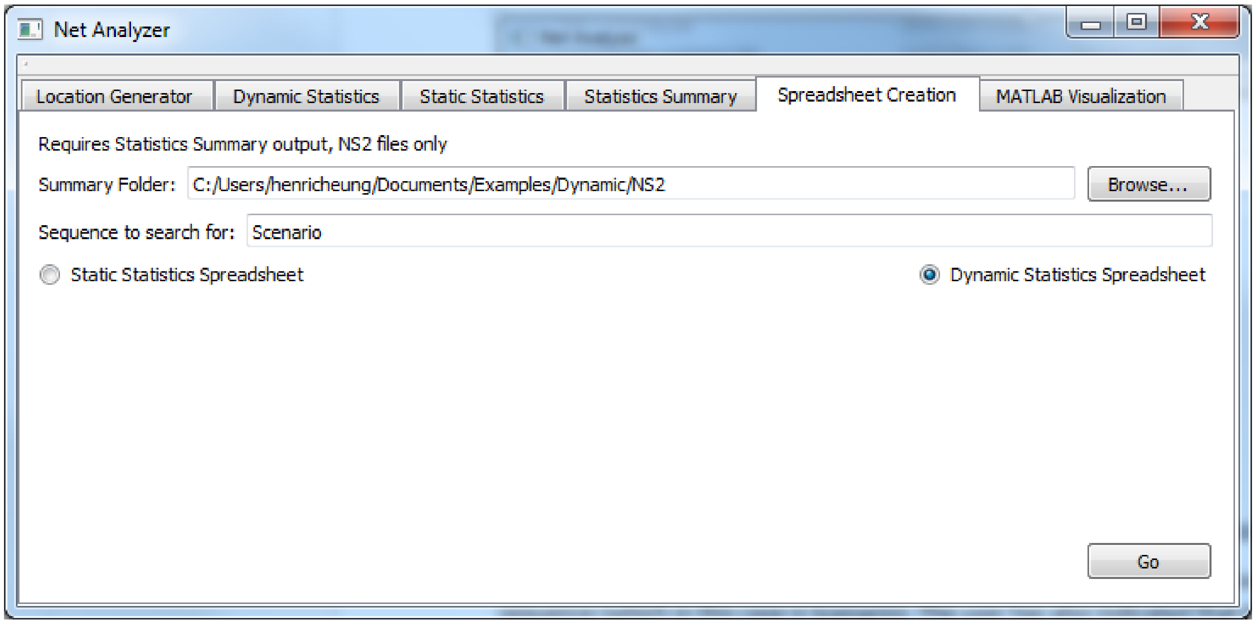
Run time (Dynamic Stats only): 150

The directory containing the output is given by the user, here. Since the scenario files are formatted as “Scenario#-#.txt”, the sequence should be <u>Scenario</u> to find all the output folders (note that this is case sensitive). The user has indicated that this will be a summary of Dynamic Statistics, and that there should be a run time of <u>150</u> seconds.

The program will then create summary statistics from all the files that match the sequence. The summary statistics will be outputted within their own folder contained within the user specified directory. For this example it will be Dynamic/NS2/Scenario.summary'.

Spreadsheet

Once the summary has been created go to the Spreadsheet Creation tab, where you will need to enter the directory, sequence and spreadsheet type.



Requires Statistics Summary output, NS2 files only

Summary Folder: C:/Users/henricheung/Documents/Examples/Dynamic/NS2

Sequence to search for: Scenario

○ Static Statistics Spreadsheet [✓ Dynamic Statistics Spreadsheet]

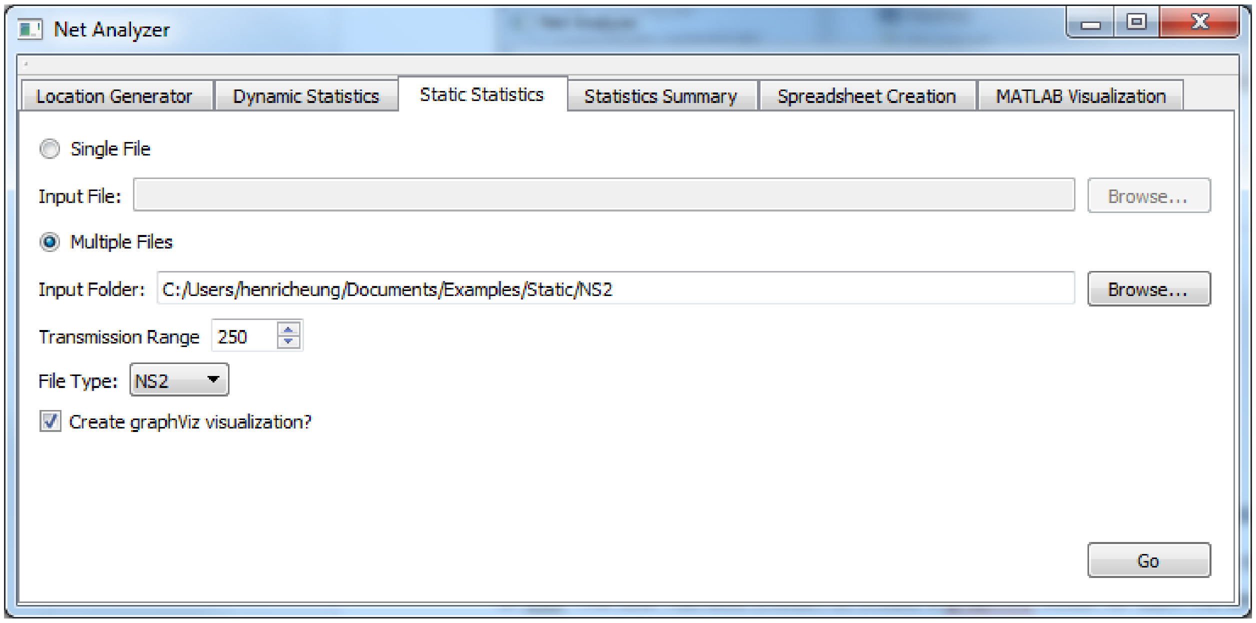
Here, the directory should be the same as the one given in the Statistics Summary, and so should the sequence (which in this case is Scenario). The user has also indicated that this is going to be a spreadsheet of Dynamic Statistics.

The program will then create an '.xml' spreadsheet for all the files used in the summary and output it to the summary folder. The file outputted for this example (Scenario.spreadsheet.xml) can be found in the example folder located in: Dynamic/NS2/Scenario.summary

Static Scenarios (NS2)

Static Statistics

Select the Static Statistics tab, where the user is then prompted to fill in the necessary parameters.



Single File

Input File:

[✓ Multiple Files]

Input Folder: C:/Users/henricheung/Documents/Examples/Static/NS2

Transmission Range 250

File Type: NS2

[✓] Create graphViz visualization?

The directory given will contain the verydense scenario files, and the user has given a transmission range of 250. The user has also chosen to create a graphViz visual for each file analyzed.

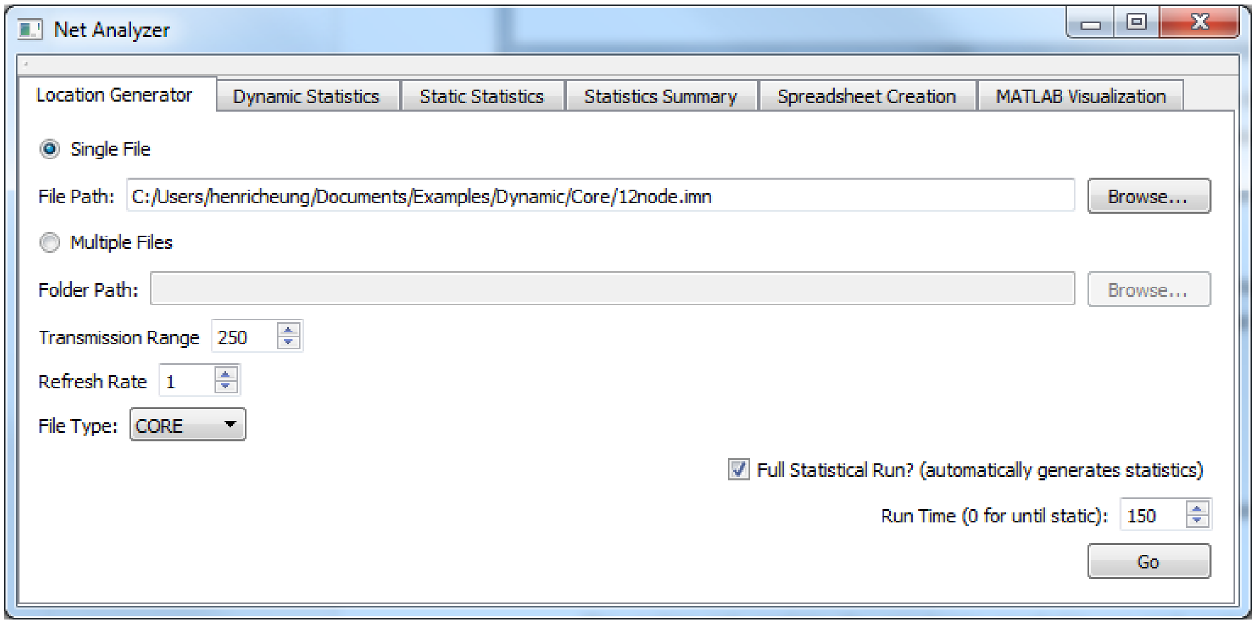
The program will then create statistics for all the scenarios and output them within their own output folders.

The user may choose to create a summary and spreadsheet for these scenarios. If so, the user must follow the exact same procedure as with HighMobility except that the user must choose static instead of dynamic when creating these files.

12node.imn (CORE)

Location Generator

Select the Location Generator tab, then click "Browse" to open a file explorer to select your .imn file



Net Analyzer

Location Generator | Dynamic Statistics | Static Statistics | Statistics Summary | Spreadsheet Creation | MATLAB Visualization

[✓ Single File]

File Path: C:/Users/henricheung/Documents/Examples/Dynamic/Core/12node.imn

Multiple Files

Folder Path:

Transmission Range 250

Refresh Rate 1

File Type: CORE

[✓] Full Statistical Run? (automatically generates statistics)

Run Time (0 for until static): 150

Make sure to select the CORE file type. In this case, the transmission range is 250 and the refresh rate is 1. The user has also chosen to collect Dynamic Statistics, with a run time of 150. Clicking "Go" will have the GUI launch NetAnalyzer.exe and start analyzing 12node.imn.

Output

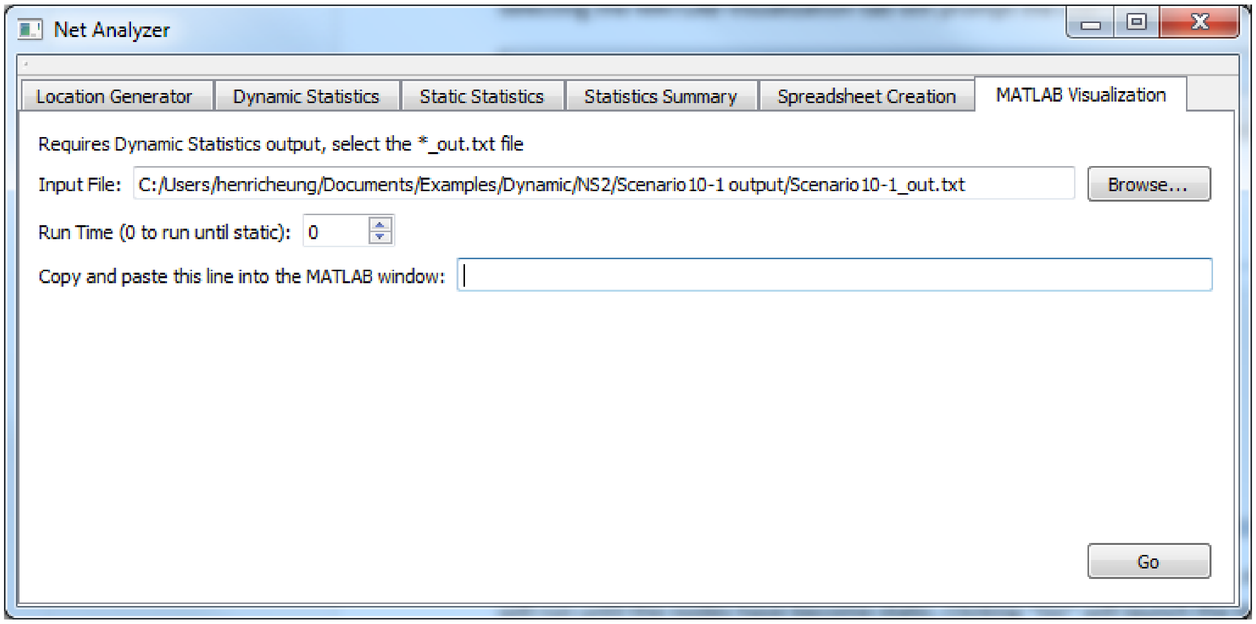
Console output can be found in out.put.

Note:

> *This section only explains the very basics needed to run each section of the program and only specifies the requirements that the user must input in order to run. See Documentation file for more specific requirements.*

Animation & Graphs

Selecting the MATLAB Visualization tab will prompt the user to give the output file, as well as a run time.


Net Analyzer

Location Generator | Dynamic Statistics | Static Statistics | Statistics Summary | Spreadsheet Creation | MATLAB Visualization

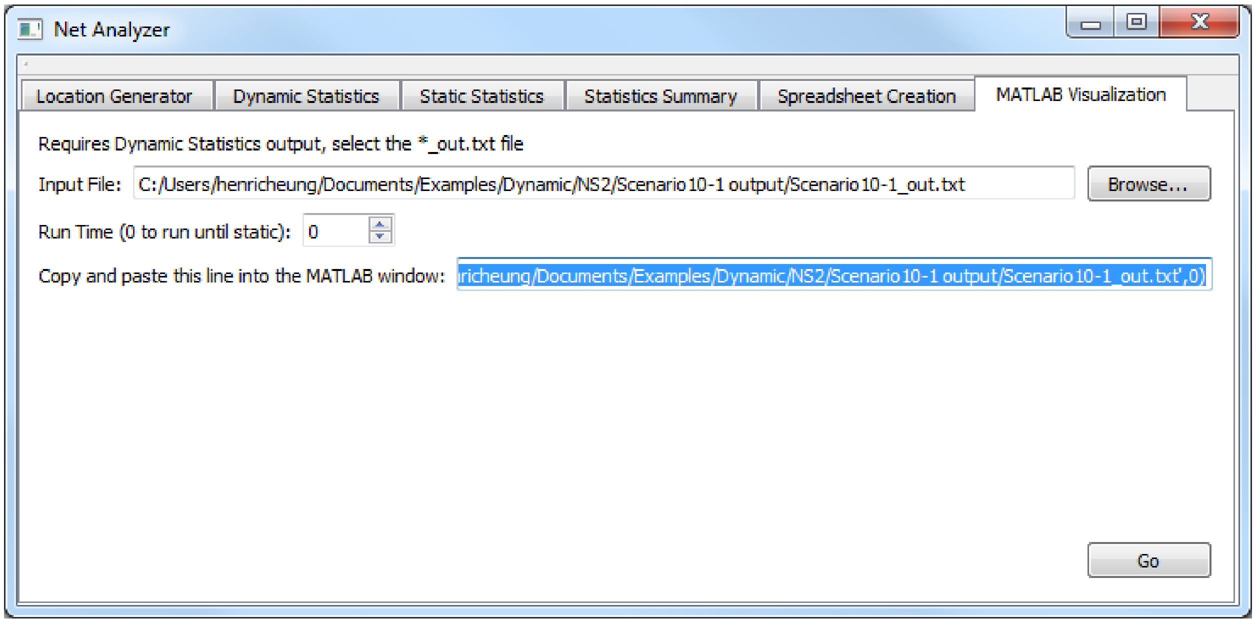
Requires Dynamic Statistics output, select the *_out.txt file

Input File: C:/Users/henricheung/Documents/Examples/Dynamic/NS2/Scenario10-1 output/Scenario10-1_out.txt

Run Time (0 to run until static): 0

Copy and paste this line into the MATLAB window:

Here, the user has selected an output file and has chosen a run time of 0, meaning that the animation will run until the nodes have become static. Clicking "Go" will launch the MATLAB command window, as well as fill the empty line with the animations_graphs() function and it's parameters.


Net Analyzer

Location Generator | Dynamic Statistics | Static Statistics | Statistics Summary | Spreadsheet Creation | MATLAB Visualization

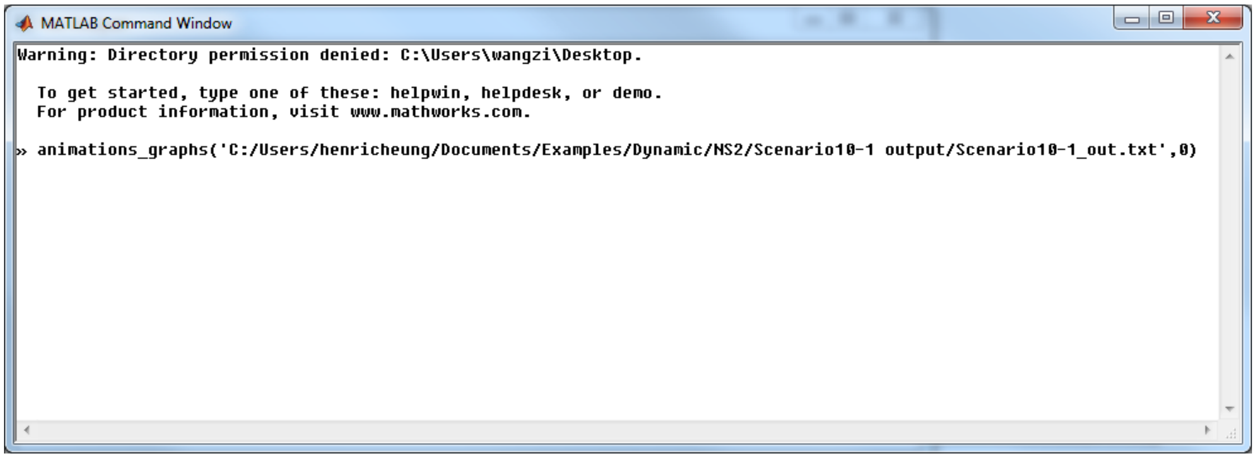
Requires Dynamic Statistics output, select the *_out.txt file

Input File: C:/Users/henricheung/Documents/Examples/Dynamic/NS2/Scenario10-1 output/Scenario10-1_out.txt

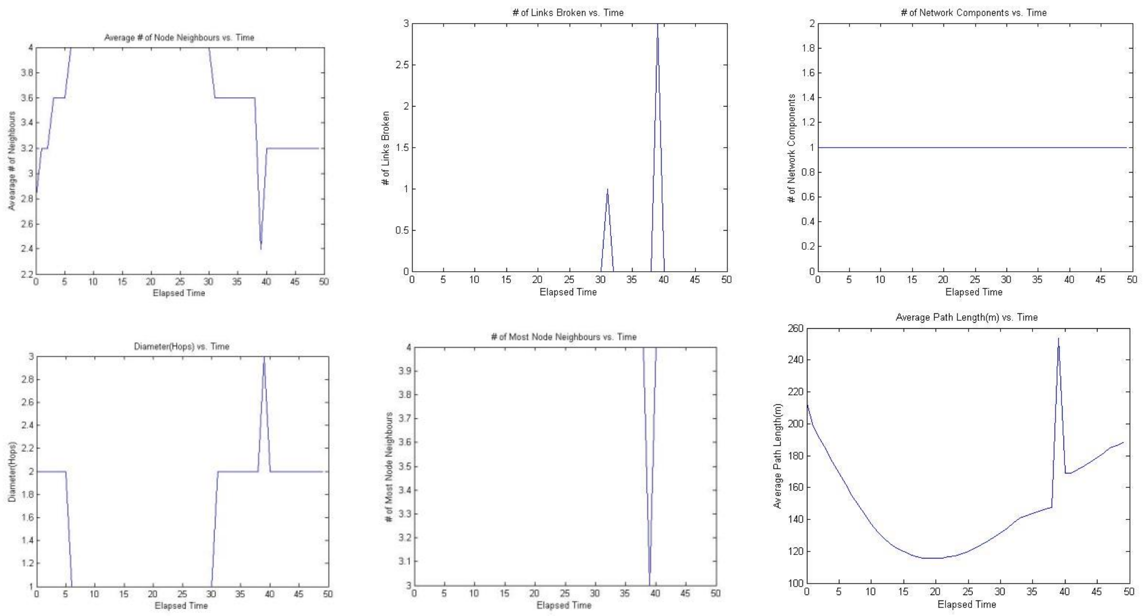
Run Time (0 to run until static): 0

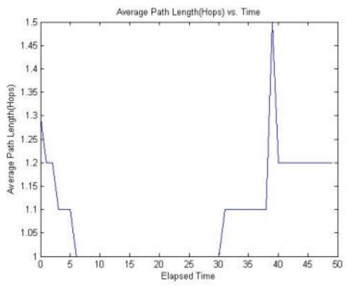
Copy and paste this line into the MATLAB window: richeung/Documents/Examples/Dynamic/NS2/Scenario10-1 output/Scenario10-1_out.txt',0)

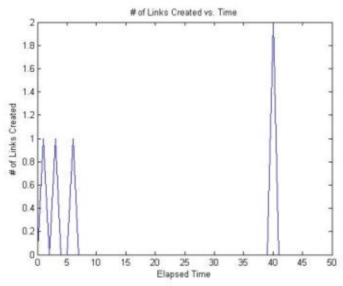
Then, the user must copy and paste the indicated line into the MATLAB command window.

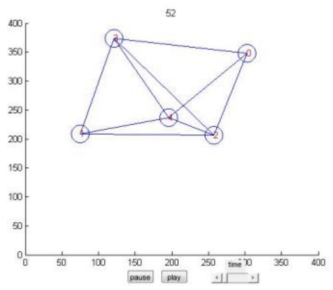


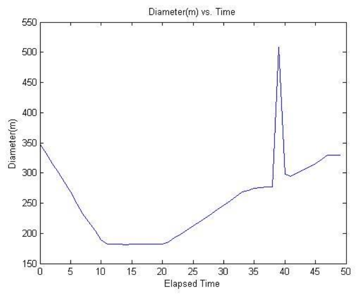
```
Warning: Directory permission denied: C:\Users\wangzi\Desktop.

  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> animations_graphs('C:/Users/henricheung/Documents/Examples/Dynamic/NS2/Scenario10-1 output/Scenario10-1_out.txt',0)
```











In.put

NetAnalyzer.exe is actually controlled by a plaintext file generated by the GUI, then passed to it through a command line parameter. If NetAnalyzer.exe is launched with more or less than one parameter, it will ask for the name of the command file. The file generated by the GUI is named in.put by default, however the file extension doesn't matter, only it's formatting. It is recommended to avoid .txt extensions, as the Net Analyzer (when running multiple file analysis on NS2 files) will attempt to analyze all .txt files, even if they are not scenario files (which can cause crashes).

The first two lines of in.put will be an integer (from 1 to 5) which represents the desired operation, and the path to the file/directory as a string.

The next lines change depending on what analysis is being performed.

For the Location Generator (1), there are 6 more lines. In order, they are:

1. Refresh rate (integer, 1-1,000)
2. Transmission range (integer, 1-10,000)
3. Full statistical run-through (true or false)
4. File extension (string)
5. Single or multiple files to analyze (string)
6. Run time, if doing a full statistical run-through (integer)

For the Dynamic Statistics (2), the next 3 lines are:

1. Run time of simulation (integer, 0-10,000, where 0 is "until static")
2. Generate GraphViz visual (true or false)
3. When to generate the GraphViz visual (integer)

For the Static Statistics (3), the next 4 lines are:

1. Generate GraphViz visual (true or false)
2. Transmission range (integer, 1-10,000)
3. File extension (string)
4. Single or multiple files to analyze (string)

For the Statistics Summary (4), the next 4 lines are:

1. Sequence to search (string)
2. File extension (string)
3. Dynamic or Static Statistics to summarize (string)
4. Run time (integer, 1-10,000) (note: this is only relevant for Dynamic Statistics)

For Spreadsheet Creation (5), there are 2 more lines:

1. Sequence to search (string)
2. Static or Dynamic Statistics Summary (string)

Example in.put

```
1
C:/Users/henricheung/Documents/Scenario10-1.txt
1
250
false
.txt
Single
0
```

This in.put tells the Net Analyzer to run the Location Generator on Scenario10-1.txt, and that the refresh rate is 1, with a transmission range of 250, NOT to run a full statistical analysis (only generating the scenario10-1_out.txt file), the file extension is ".txt" and that there is only a single file to be analyzed. Note that the GUI will produce file paths using / instead of \.

Statistics Calculated

This section will explain what each of the statistics created through the program mean and how they were calculated. (This section was originally written by Hussain Saeed and Jordan Edgecombe).

The average number of node neighbours is:

What it means: Two nodes are considered neighbours if they directly connect with each other, or in other words, if they share an edge with each other.
How it was calculated: In NS2 and Opnet, an edge is made when two nodes' coordinates are within the transmission range from one another, and in Core it is made when the two nodes share the same WLAN. With *Boost Graph Library*, a vertex iterator can be made that can loop through every unique node in the network. For each node, *Boost Graph Library* can then use an edge iterator to see how many other nodes it is directly connected to. Each connection is then recorded in a counter variable, and eventually this variable gets divided by the total number of nodes to get the average number of node neighbours.

The node(s) with the most neighbours is/are:

What it means: The nodes that share the most connections with other nodes in the network
How was it calculated: While the vertex iterator loops through every single node, the ones that have the most node neighbours are picked out and stored in a specific array. Then, after the vertex iterator is done, this array just gets outputted.

The number of nodes that the node(s) with the most node neighbours have is:

What it means: The nodes with the most node neighbours have this number of connections
How it is calculated: While the array for the nodes with the most neighbours is being created, another array is made to store the number of neighbours that the nodes with the most connections have. One element of this array then gets simply outputted for this statistic.

The average path length for all source-destination pairs (in meters):

What it means: This is the average shortest distance along the edges between any two nodes, regardless of whether or not they are directly connected to each other. To imagine what 'shortest distance along the edges' means, let's say node 1 and node 3 are not connected and that the path between them that goes through node 2 is shorter in distance than every other possible node path between them. So let's imagine that node 1 is connected to node 2 by an edge that is 200 meters long and node 3 is connected to node 2 by an edge that is 150 meters long. This program will say the distance between nodes 1 and 3 is 350 meters, no matter what the straight line distance between them truly is.
How it is calculated: When the program reads through the input file, it takes in the x and y coordinates for every node. Using these coordinates, the distances between directly connected nodes are then calculated and are saved as edge weights. Then, using a *Boost Graph Library* function called dijikstra_shortest_path and the vertex iterator again, the distance along the edges between any two nodes is calculated and added to a total-sum variable. This variable is then divided by the total number of all possible edges to get the average.

The average path length for all source-destination pairs (in hops):

What it means: This is the average minimum number of edges that exists between any two nodes.

How it is calculated: The same way as was calculated in meters, except that instead of the edge weights being the actual distances between the nodes' coordinates, all of the edge weights are set to 1.

The network diameter (in meters):

What it means: The longest 'shortest distance along the edges' between any two nodes, regardless of whether or not they are directly connected to each other.

How it is calculated: While the dijikstra_shortest_path function found the shortest distance along the edges between any two nodes, an if-statement was created that said that if the shortest path was greater than the previous longest shortest path (originally set at 0 meters), then that would become the new longest 'shortest path along edges'. After every path was calculated, the last path to become the new longest shortest path was then seen to be the greatest and was then output.

The network diameter (in hops):

What it means: The largest minimum number of edges that exists between any two nodes.

How it is calculated: The same way as calculated in meters, except that instead of the edge weights being the actual distances between the nodes' coordinates, all of the edge weights are set to 1.

Total number of components:

What it means: A component is a set of nodes that are all reachable to one another through a path of edges.

How it is calculated: This statistic is calculated simply through *Boost Graph Library*'s *connected_components* function.

Node(s) with the highest betweeness centrality:

What it means: This statistic shows the nodes that have highest amounts of shortest paths from one node to another passing through them.

How it is calculated: Using a vertex iterator again and the *Boost Graph Library* function known as *brandes_betweeness_centrality*, the betweeness centrality is calculated for every node in the network. Then the nodes with the greatest values get put into a separate array, which then gets outputted.

Links Added/Links Broken:

What it means: This statistic shows the number of links added/broken at every second of the simulation.

How it is calculated: Using Pythagorean Theorem, the distance between each pair of nodes is calculated. If this distance is less than or equal to the transmission range, then a link exists between the pair.

An array of 1's and 0's tracks where these links exist. If a link is added, then the array has a 1 at that point (ex: if node 0 and node 1 are connected, array[0][1] = 1 and array[1][0]=1). If a link is broken then a 0 is placed at that point. If an entry goes from 1 to 0, then broken links is increased by 1 and if an entry goes from 0 to 1 then added links is increased by 1.

Link Changes:

What it means: It is the amount of times a link gets broken or created for each node throughout the simulation.

How it is calculated: This is simply calculated by having an array keeps track of every link change per node by adding 1 every time a link has been added or created for that node while another integer increments itself by two's every time a link is created or broken (by two because if a link is always broken or created for two nodes).

Coverage (in %):

What it means: It is the percent of the defined area that has been passed through by a node at any point in time

How it is calculated: The area is divided into a virtual grid, represented by a two-dimensional array. During each cycle, each node increases the array value corresponding to its location by 1. The coverage percent is determined by determining how many parts of the array have a value greater than 0, then dividing that by the total parts and multiplying by 100. The final array is outputted at the bottom, giving a general impression of where the nodes are concentrated as well as allowing for depictions through MATLAB surface graphs. A separate output (surface_graph) gives a format that can be copied into MATLAB as an array for easy visualization of the node distribution.

Neighbourhood Instability:

What it means: It is the approximate rate at which the nodes create and break links.

How it is calculated: The instability is calculated for each node by determining how many links were created and broken with that node, divided by the number of neighbours it has at that point in time. The output is given in 10-cycle periods, starting with the 0-9 period. A final value giving the average instability of all nodes over the entire period is listed at the bottom. A separate output (average_instability) gives the average of the nodes at each point in time.

Clustering Coefficient:

What it means: It is the ratio of radio links among neighbours and the number of neighbours.

How it is calculated: The clustering coefficient is calculated for each node by determining how many links there are between neighbours of the node, divided by the number of neighbours it has at that point in time. A final value giving the average coefficient of all nodes over the entire period is listed at the bottom. A separate output (average_clustering) gives the average of the nodes at each point in time.

The Wikipedia page http://en.wikipedia.org/wiki/Clustering_coefficient provides a decent summary of how exactly this is determined.

Degree of Randomness/Predictability:

What it means: It is a value indicating the transition probability between states.

How it is calculated: The area is divided into a virtual grid (similar to coverage, but less segments), each part of which has a separate subgrid surrounding it (represented by a 4D array). Whenever a node moves, it records the starting and ending grid location, and increments values in the corresponding main grid (starting) and subgrid (ending). At the end, it tallies up the grid values and determines the chance of a node going from any given grid segment to another one, then uses the equation $H = -(p_i Q_{ij} ln Q_{ij})$, where p_i is the probability to stay at the starting location i and Q_{ij} is the probability to transition from state i to state j. The H values for each segment of the main grid are summed up and averaged, giving the final output.

Issues or Help

See the *Documentation* file for more specific information on the different aspects of the program.