



ARL-TR-7556 • DEC 2015



Tools and Methods to Create Scenarios for Experimental Research in the Network Science Research Laboratory (NSRL)

By Anjuli R Smith and Rommie Hardy

Approved for public release; distribution unlimited.

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.



Tools and Methods to Create Scenarios for Experimental Research in the Network Science Research Laboratory (NSRL)

by Anjuli R Smith and Rommie Hardy
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
<p>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY) December 2015		2. REPORT TYPE Final		3. DATES COVERED (From - To) 06/2015	
4. TITLE AND SUBTITLE Tools and Methods to Create Scenarios for Experimental Research in the Network Science Research Laboratory (NSRL)				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Anjuli R Smith and Rommie Hardy				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-CIN-T 2800 Powder Mill Road Adelphi, MD 20783-1138				8. PERFORMING ORGANIZATION REPORT NUMBER ARL-TR-7556	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Using a 3-phase approach, we illustrate how to create a scenario that can be used in the Network Science Research Laboratory (NSRL) for experimental research purposes. In the first phase (Planning), the overall design of the scenario is formed; in the second phase (Execution), methods for realizing the design plan are discussed, and in the final phase (Visualization), tools and methods for visualizing and running the scenario are outlined. Although this report does not go into the full length of the work it takes to implement every permutation possible within each phase, examples are given of existing code that has been used in NSRL in order to create working scenarios. Through the use of the tools and methods outlined in this report, scenarios can be developed for experimenting research areas in network science.					
15. SUBJECT TERMS scenario creation, emulation environment, NSRL					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 24	19a. NAME OF RESPONSIBLE PERSON Rommie Hardy
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (Include area code) 301-394-1189

Contents

List of Figures	iv
1. Background	1
2. Planning Phase	1
3. Execution Phase	3
3.1 Generate XML Files	3
3.1.1 Motion.xml File	3
3.1.2 Network.xml File	4
3.1.3 Vizualization.xml File	4
3.1.4 Manifest.xml File	5
3.2 NMF	6
4. Visualization Phase	7
4.1 SDT3D	7
4.2 CORE	8
5. Conclusion	10
6. References and Notes	11
Appendix. Code Examples	13
List of Symbols, Abbreviations, and Acronyms	16
Distribution List	17

List of Figures

Fig. 1	Google Earth toolbar	2
--------	----------------------------	---

1. Background

The Network Science Research Laboratory (NSRL) at the US Army Research Laboratory (ARL) supports investigation of traditional wireless networking challenges as well as more general network science research issues. NSRL is capable of doing this by providing a controlled and repeatable emulation environment where the link and physical layer connectivity can be modeled in real time. In modeling the real-time mobility aspect of the emulated environment, the development and creation of scenarios play an integral part. By creating scenarios that model certain mobility patterns and Army maneuvers, researchers can study how the network reacts under certain conditions and reduce unknown factors thereby focusing solely on how the algorithm or protocol being tested performs. In this report, the tools and methods used for developing a scenario are discussed and an example scenario is shown. The kinds of scenarios we are creating can be run in multiple visualization tools such as the Scripted Display Tool 3D (SDT3D) and the Common Open Research Emulator (CORE).

The process of developing a scenario can be examined using the following 3 phases: planning, execution, and visualization. During the planning phase, the overall area of operation is determined and areas of interest are designated along with paths to specify movement. In the execution phase, the plan is realized by adding node elements along paths or within the areas of interest outlined in the planning phase. Output from the execution phase feeds into a number of different tools that can be used to visualize the scenario during the visualization phase. We examine these 3 phases in detail by describing the creation of a scenario based upon a vignette from the Multi-Level Scenario 1.0 (MLS) documentation produced by the US Army Training and Doctrine Command (TRADOC).


2. Planning Phase


To begin creating a scenario, we start by documenting a technical narrative that outlines certain important factors. One should take into account typical Army mobility in a design and make the maneuvers and outcome as realistic as possible. One tool that has proven useful during this phase is Google Earth. Using Google Earth, a keyhole markup language (KML) file is created, which contains all the locations and paths specified in the documentation. Using the MLS 1.0 documentation as an example, the location of where groups of nodes are present and the paths they use to get from place to place can be determined. These locations can then be identified and marked using the tools in Google Earth. The tools can be found in the toolbar above the map and look like the picture in Fig. 1. These tools


are used to set placemarks, draw polygons, add paths, and include image overlays. (Note: The camera feature is not used in the creation of scenarios).




Fig. 1 Google Earth toolbar

To set a placemark, click the yellow pushpin  on the toolbar. A pushpin will appear on the map along with a window to edit its settings. If the pushpin is used to represent a position above ground level, increase the altitude. If the placemark is not in the correct position, you can drag the pushpin to move it into the corrected position. Give the placemark a name that describes the location that it represents and click OK.

Generally, polygons  are useful for mapping an area where a group of nodes are performing different maneuvering operations like wandering, meandering, or becoming stationary. The edges of the polygon become the boundary in which the nodes perform those operations. To create this, click the polygon feature, then click on the screen. You do not need to drag, just click along the edges of the area that you want to cover. It may be helpful to adjust the transparency in the polygon settings so that you can see the area underneath the polygon.

The path feature  defines a specific route that an individual or group will be taking to get from point A to point B. This is most useful for tracing roads or sidewalks so that the nodes' movements are more realistic (i.e., they are not walking through rivers or buildings). These movements, along with what the nodes' appearance and network connectivity, are defined in the extensible markup language (XML) files, which we create next.

The image overlay button  can be used to add heat maps. When creating these, you need to have a link to the image you want to overlay onto the map. Start by clicking the icon. Green markings outlining a square shape will appear on the screen, along with the settings box. In the settings, click browse to add in the image you are overlaying into the map. Next, click and drag the edges and corners of this box to adjust its size. You can also drag the little green diamond on the box to rotate the image.

To save, right click the folder on the left panel containing the new places (this should be the temporary places folder. If it is not, move all the places to the

temporary folder) and select Save Place As. Save the file as a .kml file for use in the execution phase.

3. Execution Phase

3.1 Generate XML Files

In this phase of scenario generation, multiple XML files are created that contain all the information needed to create your scenario. The easiest way to create any of these files is to take the files of an example scenario and change its parameters to suit your needs.

In order to construct the framework for a scenario, the following 4 files need to be created: motion.xml, network.xml, visualization.xml, and manifest.xml. The motion.xml file defines the movements each node will take with respect to the locations previously defined in Google Earth. The network.xml file consists of defining network groups, then assigning these networks to the nodes defined in the motion file. The visualization.xml file changes the appearance of the nodes, and finally, the manifest.xml file brings all the files together to run within the Network Modeling Framework (NMF).

3.1.1 Motion.xml File

In a motion.xml file, movements can be defined for a single node or a group of nodes. The different kinds of movements are defined in the NMF documentation; some common movements are stationary, wander, and path. To define movement for a group of nodes, use the <repeat> tag. An example of how to perform the repeat tag can be found in the code in Section A-1 of the Appendix.

While syntactically unnecessary, using a prefix helps to visualize which nodes are in the same group and is helpful when referencing the nodes for modifying visualization and adding networks. Using triggers allows you to specify when you want a motion to occur during the scenario. As a result, this allows you to assign multiple motions to each node.

Here is a portion of the event plan, which contains the delay tag referenced by the trigger tag:

```
<event type="periodic">
  <parameter name="initial">0</parameter>
  <parameter name="period">1200</parameter>
  <parameter name="limit">-1</parameter>
  <throws delay="00">CrossLineOfDeparture</throws>
  <throws delay="60">BCompanySeizesObjective</throws>
  <throws delay="75">ACompanyEncountersPatrol</throws>
</event>
```

3.1.2 Network.xml File

The network groups are created in the network.xml file. We start by defining the networks with a name and different parameters such as bandwidth and range:

```
<network name="ACompany">
  <parameter name="addressPool">10.0.1.0/24</parameter>
  <parameter name="type">localRadio</parameter>
  <parameter name="frequency" units="GHz">2.400</parameter>
  <parameter name="txpower" units="dBm">20</parameter>
  <parameter name="noisefigure" units="dB">4</parameter>
  <parameter name="bandwidth" units="MHz">10</parameter>
  <parameter name="range" units="m">.001</parameter>
  <parameter name="connectivityStyle">distance</parameter>
</network>
```

Then we assign these networks to the nodes:

```
<!-- A Company -->
<repeat name="X" range="node002-node020">
  <node name="$X$">
    <interface name="rad0" net="ACompany"> </interface>
  </node>
</repeat>
```

A node doesn't need to be assigned a network, but those that are can be assigned multiple networks as long as each network is on a different interface for that node.

3.1.3 Vizualization.xml File

To aide in visualizing the nodes, which is discussed in Section 4, a visualization.xml file can be created. This file defines the color, size, and shape of the nodes that are used in the overlay of the visualization tool. Without defining the nodes using a visualization file, nodes are automatically created as red spheres on

the map; however, if you want to change the nodes from the default red sphere, you can modify the XML file to edit the shape and color of the nodes. Another option is to assign an image, or sprite, for the node. A sprite is a method in computer graphics terms for seamlessly integrating a flat image onto a 3-dimensional (3D) layer. To incorporate sprites into the scripted display tool (SDT), it has to be manually added into the .sdt file, which is explained later. If you choose not to include a visualization.xml file, make sure it is not loaded from within the manifest.xml file. The following example code shows how to change the color of nodes to blue within the visualization.xml file:

```
<!--Battalion-->
<repeat name="N" range="node001-node061">
  <node name="$N$">
    <parameter name="color">0000ff</parameter> <!--Blue-->
  </node>
</repeat>
```

3.1.4 Manifest.xml File

Finally, the manifest.xml file must be created. This file can be particularly lengthy and can get confusing since it can encompass all or a lot of the various different command line options that exist within the NMF tool. This includes loading input files, changing the parameters values within those files, generating output file and changing the various parameters for each of the desired outputs. Starting from a previously created manifest file and modifying it to suit your needs is the best method of creating this file. Example files are provided in the NMF software release. The first command that is normally given in the manifest file is to load the motion, network and then visualization files. Here is what the code looks like to load the motion file:

```
<action name="Load and Compile Motion">
  <action type="load">
    <!-- Searches manifest folder, then application working folder -->
    <!-- This loading will be passed in with variables attached -->
    <file>scenario1.motion.xml</file>
  </action>
  <!-- Triggers the compilation of motion -->
  <action type="execute">
    <mode>motion</mode>
    <parameter name="endTime">1200</parameter>
  </action>
</action>
```

Next, we need to write out the command for the generation of different files, such as Emulation Event Log (.eel) and SDT. Here is an example of what the block of code for the .eel file should look like:

```
<action type="generate">
  <mode>eel</mode>
  <directory base="application">Squad_deployed</directory>
  <prefix>nems</prefix>
  <parameter name="startTime">0</parameter>
  <parameter name="endTime">800</parameter>
  <parameter name="reportingInterval">1</parameter>
  <parameter name="namingType">nem</parameter>
  <parameter name="locationEnabled">true</parameter>
  <parameter name="linksEnabled">true</parameter>
</action>
```

Here is the .sdt file:

```
<action type="generate">
  <mode>sdt</mode>
  <directory base="application">home_preview</directory>
  <parameter name="startTime">0</parameter>
  <parameter name="endTime">1440</parameter>
  <parameter name="reportingInterval">1</parameter>
  <parameter name="timeScaler">3</parameter>
  <parameter name="namingType">Interface</parameter>
  <parameter name="locations">true</parameter>
  <parameter name="links">true</parameter>
  <prefix>scenario1.preview</prefix>
</action>
```

3.2 NMF

Now that we have all the XML files created, we can use the US Naval Research Laboratory's (NRL) NMF software to create the scenario as .sdt and .eel files. Start by putting the KML file, XML files you created, any pictures, and all of the resources.xml files in the same folder. Then, put that folder into the same directory as the NMF executable and its related files. Finally, to run NMF, open a terminal, change to the directory that contains the NMF executable, and enter the following command:

```
mono NMF.exe -m.
```

This lists all the manifest files within this directory. Select the number that corresponds to the manifest file that you wish to run. This generates the files that you specified NMF to create in the manifest file.

4. Visualization Phase

4.1 SDT3D

A quick and simple way to visualize the mobility being performed is to create an .sdt file. These files can be run using the SDT3D, which is available from <http://downloads.pf.itd.nrl.navy.mil/sdt/>. SDT3D uses the National Aeronautics and Space Administration's (NASA) World Wind application programming interface (API) to create interactive visualizations of 3D globe, map, and geographical information along with a set of overlaid nodes. Node are automatically visualized as red spheres of the map; however, if you want to change the nodes from the default red sphere, you can create an XML file to edit the shape and color of the nodes. Another option is to assign an image or sprite for the node. This has to be manually added into the .sdt file, which is explained later. If you choose not to include a visualization.xml file, make sure it is not included in the manifest.xml file. In this example, the color of the nodes are changed to blue:

```
<!--Battalion-->
<repeat name="N" range="node001-node061">
  <node name="$N$">
    <parameter name="color">0000ff</parameter> <!--Blue-->
  </node>
</repeat>
```

To add sprites (2- or 3-dimensional image) to an .sdt file, open the .sdt file in a text editor and make changes so that the file incorporates the locations of the sprites and the relationship to each of the nodes in the .sdt file. An example is shown in the following sample code excerpt:

```
path "home_preview/home_preview/sprites/../../home_preview/../../home_preview/sprites/"

sprite soldier image soldier.gif

clear nodes
node node001.rad0 label off type soldier
node node001.rad0 position -107.869881106542,37.2003840799984,0
```

4.2 CORE

CORE is a tool for emulating networks. To run the scenario in CORE, we need to create an .imn file, configure the Extendable Mobile Ad hoc Network Emulator (EMANE), and then run the .eel file. The first step, creating an .imn file, sets up a canvas containing routers that correspond to the nodes in the .eel file you have. These routers will receive messages from EMANE as specified in the .eel file when you run the emulation. To start, open CORE.

After opening CORE, add nodes by clicking on the round blue router on the left panel of CORE. Select the MDR. From here, there are 2 different ways to add nodes onto the canvas. Either click the canvas to individually place nodes onto it, or click on Tools > Topology Generator > Random and select the number of nodes that is closest to the number you need. After the nodes appear on the canvas, either locate the nodes you don't need and delete them or click the canvas to increase the number of nodes. Now that we have nodes, we need to create an EMANE cloud so that CORE can receive messages from the .eel file through EMANE.

Click on the square blue Ethernet hub on the left panel. Select the cloud (which is a wireless local area network [LAN]) and place it on the canvas by clicking anywhere on the white grid. Right click the cloud, select configure, and press the button labeled link to all nodes. In the same window, click the EMANE tab and select 1 of the EMANE options, like bypass. This makes the cloud an EMANE connection and since the cloud is connected to all nodes, these nodes will mirror the movement fabricated in the .eel file.

To create additional network connections among the nodes, you can choose to create more wireless networks by adding more clouds, or you can create a wired connection between 2 nodes. To make a wireless connection, add another cloud and link it to the desired nodes. Unlike before, do not make this one an EMANE cloud. Instead keep it as a basic cloud and adjust the range, bandwidth, delay, loss, and jitter as necessary to your specifications. To make a wired connection, click the black line icon on the left panel. Then sequentially click the 2 nodes that will have the connection. Next, we need to make sure CORE's EMANE parameters are set correctly so that it is listening to EMANE.

Open the nems.eel file to see the value at which the node numbering starts. Then, click on EMANE options > NEM Parameters and change the starting network emulation module (NEM) to the observed value.

Next, open the eventservice.xml file and take note of the value for the eventservicegroup and eventservicedevice parameters. (If you don't have the file, there is an example of one in Section A-2 of the Appendix along with the

eelgenerator.xml file [Section A-3]). Under “Platform Attributes” in the EMANE options, compare the value of the event service group and event service device to those in the eventservice.xml file. If they are not the same, change the values in the .xml file to match.

Finally, we need to edit the canvas settings to complete the EMANE configuration. Select Canvas > Size/scale and start by setting the altitude to 0 m. The next steps depend on the scenario you have created. Start by looking at the KML file in Google Earth and locating the approximate northwest corner of where your nodes will be moving and copy down the latitude (lat) and longitude (lon) coordinates (or look at the nems.eel file if that is easier). In the canvas settings under reference point, set the lat and long coordinates to those that you just obtained. The size and scale setting will need to be adjusted as you see fit. If the scenario takes place over a large expanse of land, then you will need to make the meters-to-pixels ratio larger by increasing the number of meters in 100 pixels, and you also may need to increase the size of the canvas by increasing the number of pixels in the width and height. Save this as an .imn file.

Now that we have a canvas set up, we can run the nems.eel file and have that movement emulated in CORE. First, put the following files together in the same directory: eventservice.xml, eelgenerator.xml, and nems.eel. Next, edit the file core.conf, which should be found in the /etc/core directory. In the file is a parameter called emane_event_monitor. Change its value to true (note: altering this file will most likely require root privileges). After doing this, reset the core daemon by entering the following command into a terminal:

```
sudo services restart core-daemon.
```

This next step is only for those who are running CORE from a machine other than vcore. If you simply downloaded the CORE application, then you will also need to download EMANE and Quagga separately, then reset the core daemon. There are many more things that you can change within CORE to fit your scenario and other errors and problems you may run into that we haven’t addressed. For more information regarding CORE, see the CORE documentation.¹

It’s finally time to run the .eel file. In a terminal, change directories to where the .eel and .xml files are located. Make sure that you have CORE open, the .imn file for this scenario loaded and the green play button on the left panel pressed so the nodes are ready to receive motion messages. In the terminal, run the following command:

```
emaneeventservice eventservice.xml -l 4
```

Going back to the CORE window, you should now see the nodes moving in the expected pattern. If they are not, check the bottom right corner of the CORE window for a yellow engine icon. If it is there, click it and you will be able to view the error messages. If you do not see the engine icon, then CORE is unable to communicate with EMANE and you will need to figure out why they are not connecting.

Once you have the nodes moving, you can analyze the network connectivity between them by setting up traffic flows and trace routes. Traffic can also be sniffed using tcpdump and Wireshark.

5. Conclusion

The tools and methods described in this report have been used to create multiple scenarios that are being used in NSRL. This report shows how those 3 phases along with the tools and methods used in each phase can be used to illustrate the design, execution, and visualization a scenario. Using this 3-phase approach provides an effective way to create mobile and/or static scenarios for research in network science related areas.

6. References and Notes

1. CORE Documentation, Release 4.8, June 5, 2015 [accessed 2015].
http://downloads.pf.itd.navy.mil/docs/core/core_manual.pdf.
2. Multi-Level Scenario Module 1: 7th Division, Operational scenario documentation, rev. ed. Jun 2006-May 2007, June 6, 2015 [accessed 2015].
<http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA520478>.

INTENTIONALLY LEFT BLANK.

Appendix. Code Examples

A-1 Repeat Tag

```
<!--ACompany-->
<repeat name="N" range="node001-node010">
  <node name="$N$">

    <point ref="AStart" />

    <motion type="path">
      <parameter name="pathName">APath</parameter>
      <parameter name="forwards">true</parameter>
      <parameter name="maxSpeed">280</parameter>
      <parameter name="speed">280</parameter>
      <trigger>CrossLineOfDeparture</trigger>
    </motion>
    <motion type="path">
      <parameter name="pathName">ACToIED</parameter>
      <parameter name="forwards">true</parameter>
      <parameter name="maxSpeed">280</parameter>
      <parameter name="speed">280</parameter>
      <trigger>BCompanySeizesObjective</trigger>
    </motion>
    <motion type="path">
      <parameter name="pathName">IEDToBridge</parameter>
      <parameter name="forwards">true</parameter>
      <parameter name="maxSpeed">280</parameter>
      <parameter name="speed">280</parameter>
      <trigger>ACompanyLeavesIEDSite</trigger>
    </motion>
  </node>
</repeat>
```

A-2 eventservice.xml

Code for eventservice.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventgenerator SYSTEM "file:///usr/share/eman/dtd/eventgenerator.dtd">
<eventgenerator library="eelgenerator">
  <param name="inputfile" value="diablo.eel" />
  <paramlist name="loader">
    <item value="commeffect:eelloadercommeffect:delta"/>
    <item value="location,velocity,orientation:eelloaderlocation:delta"/>
    <item value="pathloss:eelloaderpathloss:delta"/>
    <item value="antennaprofile:eelloaderantennaprofile:delta"/>
  </paramlist>
</eventgenerator>
```

A-3 eelgenerator.xml

Code for eelgenerator.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventservice SYSTEM "file:///usr/share/eman/dtd/eventservice.dtd">
<eventservice>
  <param name="eventservicegroup" value="224.1.2.8:45703"/>
  <param name="eventservicedevice" value="lo"/>
  <generator name="EEL Generator" definition="eelgenerator.xml"/>
</eventservice>
```

List of Symbols, Abbreviations, and Acronyms

3D	3-dimensional
API	application programming interface
ARL	US Army Research Laboratory
CORE	Common Open Research Emulator
EEL	Emulation Event Log
EMANE	Extendable Mobile Ad hoc Network Emulator
KML	keyhole markup language
LAN	local area network
lat	latitude
lon	longitude
MLS	Multi-level Scenario
NASA	National Aeronautics and Space Administration
NEM	network emulation module
NMF	Network Modelling Framework
NSRL	Network Science Research Laboratory
SDT	Scripted Display Tool
SDT3D	Scripted Display Tool 3D
TRADOC	US Army Training and Doctrine Command
XML	extensible markup language

1 DEFENSE TECHNICAL
(PDF) INFORMATION CTR
DTIC OCA

2 DIRECTOR
(PDF) US ARMY RESEARCH LAB
RDRL CIO LL
IMAL HRA MAIL & RECORDS
MGMT

1 GOVT PRINTG OFC
(PDF) A MALHOTRA

3 US ARMY RESEARCH LAB
(PDF) RDRL CIN T
B RIVERA
AR SMITH
R HARDY

INTENTIONALLY LEFT BLANK.