Software Requirements

Specification

for

Ad-hoc Wireless Network Simulator

Version 1.1

Prepared by Team Group 1:
Greg Moser, James Woo, Matt Roberts and Yali Zhao

Hood College Computer Science Department

November 1st 2010

Table of Contents

Revision History

Version	Reason for Change	Date
1.0	Reason: Initial version Modified By: Group 1	09/28/2010
1.1	Reason: Updated according to review comments and suggestion. Added mock-ups and updated use cases. Modified By: Group 1	10/13/2010

1. Introduction

1.1 Purpose

The software requirements specified in this document describe the Ad-hoc Wireless Network Simulator developed by Group 1. Group 1's simulator is based on the AODV Simulator [1]. The scope of this document is to describe front-end GUI requirements and back-end system component requirements.

1.2 Document Conventions

PF<Product Feature> - #.#.#.#
FR<Functional Requirement> - #.#.#.#

Each # symbolizes a level of detail. Each subsequent level is one step deeper into the details of that feature or requirement.

1.3 Intended Audience and Reading Suggestions

This document is primarily for use by stakeholders of the simulator. Its distribution is unrestricted and as such may be used by anyone for educational purposes outside the scope of this project.

1.4 Project Scope

This project's purpose is to provide a GUI demonstration of an ad-hoc wireless network simulation that exercises specific protocols using different scenarios and presents results in a user friendly format. The simulator's development will be based on the AODV Simulator [1] and will be constructed by Group 1 at Hood College's Computer Science Department. Once completed the simulator will be made available to the public for educational purposes and further research.

1.5 References

1. AODV Simulator. Ali Dabirmoghaddam, Masoud Moshref. 2010. SourceForge. 4 Oct. 2010 <http://aodvsimulator.sourceforge.net>.
2. Oracle Sun Developer Network. 2010. Oracle. 4 Oct. 2010 <http://java.sun.com/javase/technologies/desktop/>.
3. Roger S. Pressman. Software Engineering: A Practitioner's Approach. Boston: McGraw-Hill, 2009.

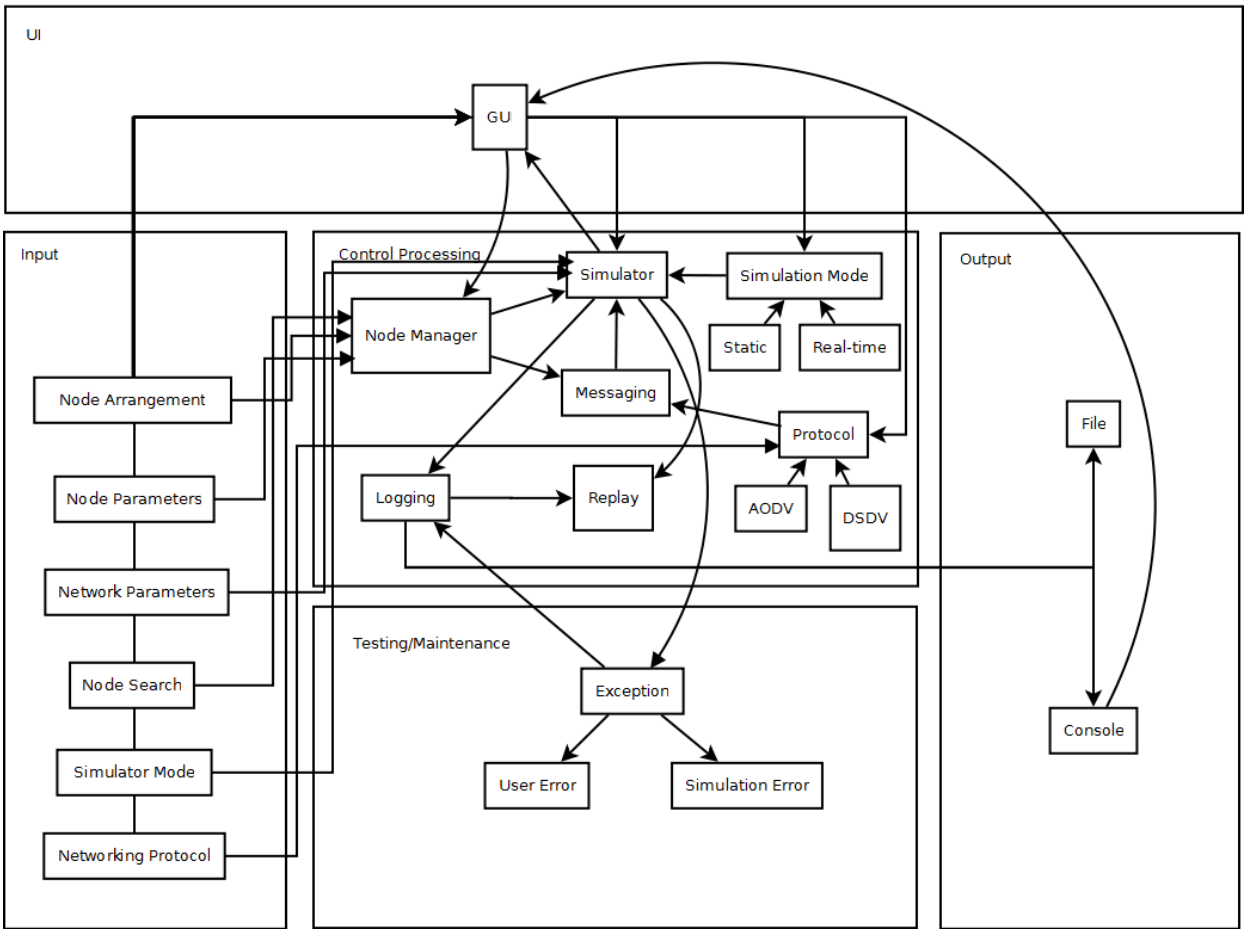
2. Overall Description

2.1 Product Perspective

The origin of the ad-hoc wireless network simulator comes from Ali Dabirmoghaddam and Masoud Moshref's Ad hoc On-demand Distance Vector (AODV) Simulator [1]. The origin will be built upon to handle the Destination Sequenced Distance Vector (DSDV) protocol as well as further enhancements and requirements described in this document.

Major components of the simulator are illustrated below in the Simulator Architecture Diagram (Fig. 2.1):

Project Architecture Diagram
Team GR1



2.2 Product Features

Req #	Description
PF-1	The simulator shall have a user-friendly and appealing GUI.
PF-2	The simulator shall be able to simulate (at the Routing layer) the Ad hoc On-demand Distance Vector (AODV) protocol and the Destination Sequenced Distance Vector () protocol.
PF-3	The simulator shall be able to accommodate any number of nodes.
PF-4	The simulator shall allow the user to change the node locations and arrangement during the simulation.DSDV
PF-5	The simulator shall allow the user to add individual nodes using drag-and-drop and also be able to move them on the display and delete them.
PF-6	The simulator shall provide the ability to modify node parameters.
PF-6.1	The simulator shall provide the capability to manually set individual node.
PF-6.2	The simulator shall provide the capability to automatically set each individual node parameter using random, but valid data within ranges.
PF-6.3	The simulator shall provide the capability to automatically set all node parameters using random, but valid data within ranges.
PF-7	The simulator shall clearly indicate to the user when the simulation begins and ends.
PF-8	The simulator shall provide detailed logging about the state of each node, the overall network environment, information about messages (i.e. size) being transferred at each increment of execution.
PF-9	The simulator shall provide text file logging capability so users can retrieve the results of their execution.
PF-10	The simulator shall provide both a textual and a graphical/animated representation of what is happening during the simulation.
PF-11	The simulator shall be developed so it is platform independent (MS-Windows, Unix, Apple is optional).
PF-12	The simulator shall be extend-able by having provisions (i.e. “code hooks”) that enable the easy integration of additional protocols.
PF-13	The simulator shall be developed in such a way that the simulator engine and the GUI are independent, allowing developers to use other GUIs.

PF-14	The simulator shall allow the user to select the overall network bandwidth and the routing protocol.
PF-15	The simulator shall enable the user to search for a node, identify it on the screen and provide its information.
PF-16	The simulator shall provide the deliver-ables needed to guide, install, run and view code.
PF-16.1	The simulator shall provide source code as a deliverable.
PF-16.2	The simulator shall provide an installation installer.
PF-16.2.1	The simulator shall provide an installation installer compatible with Windows XP/7, not just an executable file.
PF-16.2.2	The simulator shall provide an installation installer compatible with Unix on how to launch, not to compile.
PF-16.3	The simulator shall provide a brief but usable Installation Guide as a deliverable.
PF-16.4	The simulator shall provide a brief but usable User's Guide (with screen shots) as a deliverable.
PF-16.5	The simulator shall provide a brief, but usable programmer's guide as a deliverable.
PF-16.5.1	The simulator shall provide a programmer's guide with directions on how to add new protocols.
PF-16.5.2	The simulator shall provide a programmer's guide with directions on how to interchange interfaces.
PF-17	The simulator shall enable the user to execute simulation runs.
PF-17.1	The simulator shall execute simulations based on an individual node sending a message to another node.
PF-17.2	The simulator shall execute simulations based on multiple nodes sending messages to other nodes.
PF-17.3	The simulator shall execute other simulation modes according to the AODV and DSDV protocols.
PF-18	The simulator shall be released under the GNU GPLv3 license with the intention to be distributed freely for educational use.
PF-19	The simulator shall be able to replay executed simulation runs.

2.3 User Classes and Characteristics

The main user class that is anticipated to utilize the ad hoc wireless network simulator are students who can use the simulator for educational purposes. This student class of users will use the simulation to better understand wireless networks and/or to use the source code to further enhance the simulator.

Other possible, but a less important user class, could be companies who are looking for a wireless network concept and would like to use the simulator to train or teach their employees about ad hoc wireless networks.

2.4 Operating Environment

The simulator shall be developed in a platform independent manner. The simulator's code base is in Java which is a platform independent programming language. In addition, an install wizard shall be delivered to allow users to install and operate the simulator on typically Windows, Unix or an Apple OS. To successfully run the simulator, the hosting OS will need to have a Java Run-time Environment (JRE) installed, preferably JRE version 5 or newer. JREs come pre-installed on over 90% of all desktop systems on a wide range of operating systems [2].

2.5 Design and Implementation Constraints

Complex simulations, such as those using hundreds of nodes, could be limited to the hosting computer's processing power. Performance may be hindered if the simulation becomes more complex.

Development of the graphic user interface (GUI) is limited to Java Swing [2] and its capabilities. Java Swing is a desktop Java technology that can be used to create portable client applications and applets.

Since the simulator will be developed for strictly educational purposes continual maintenance will not be needed. Once the base requirements have been completed and the simulator is operational, it will be purely a voluntary effort to continue to enhance and maintain the product.

2.6 User Documentation

As specified in requirements #, # and # the following documentation components shall be delivered as formatted Microsoft Word documents, plain textual format and, if applicable, Hyper Text Markup Language (HTML) format:

- The simulator shall provide a brief but usable Installation Guide as a deliverable.
- The simulator shall provide a brief but usable User's Guide (with screen shots) as a deliverable.
- The simulator shall provide a brief, but usable programmer's guide as a deliverable.

2.7 Assumptions and Dependencies

The project could be drastically effected by the timeline available to actually develop the application. According to three point estimation [3] calculations, it is estimated to take 3.5 months to complete development on the simulator product.

Calculations:
n months
5 people
Productivity = 3450 / (5n) Lines Of Code / person-month
n months = near 3.5 months

Although, according to the available timeline, the development team only has a few weeks of development time. To compensate for the shorter timeline, the development team has decided to build upon an existing product [1], rather than starting from scratch. The above calculation's result of 3.5 months was based on developing the product from ground up.

Building upon an existing product may save time, if the existing code is well coded and documented, if it is not, then this could actually extend the timeline further.

3. System Features

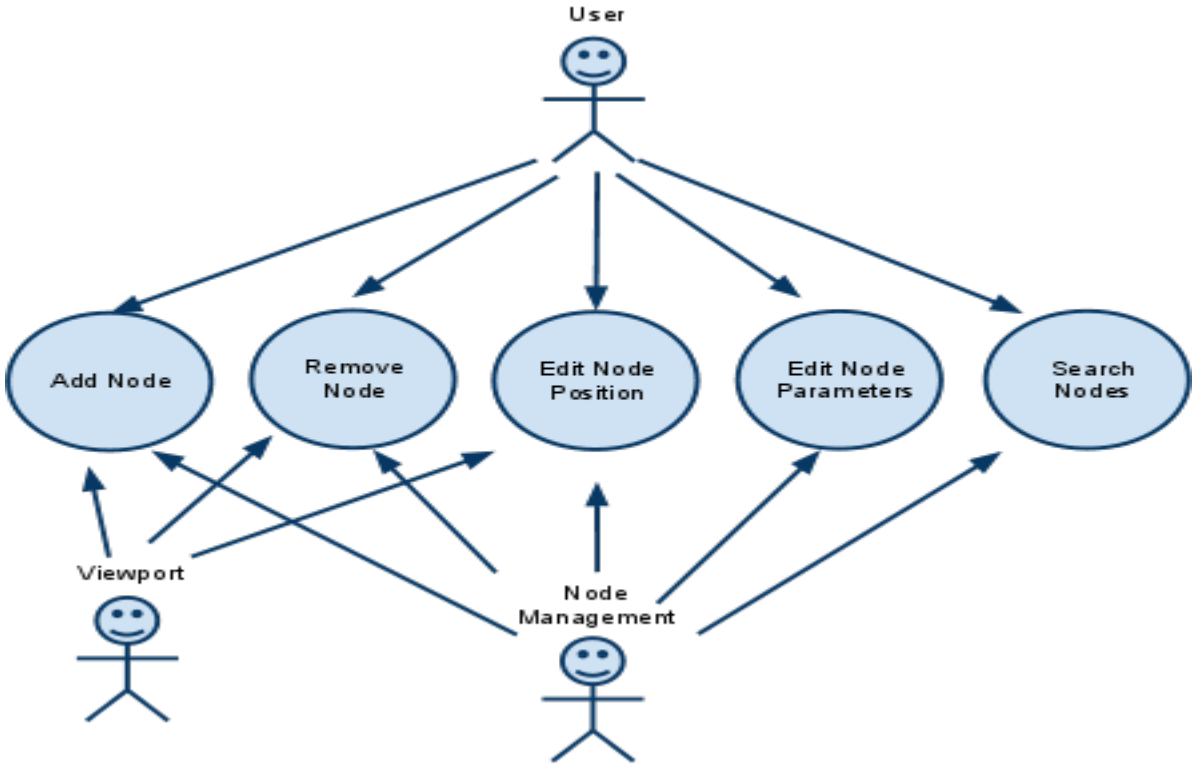


Fig. 3a: Use case diagram for the viewport and node management.

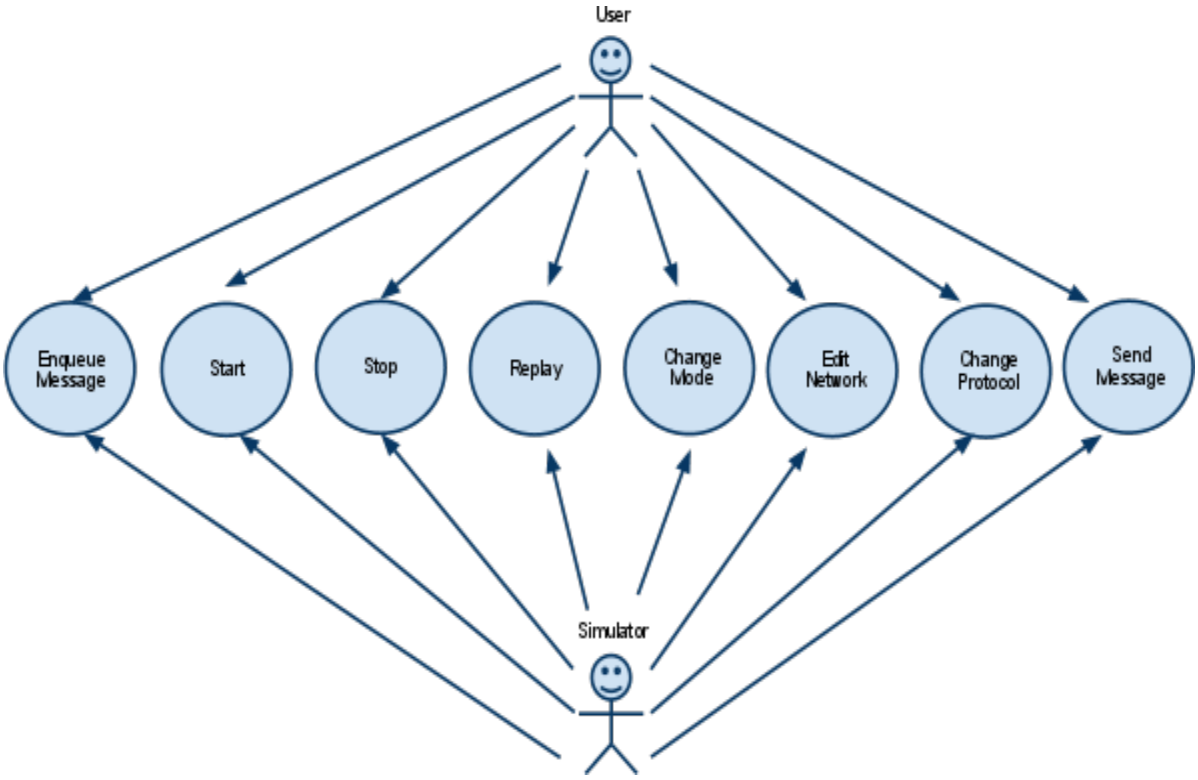


Fig. 3b: Use case diagram for the simulator.

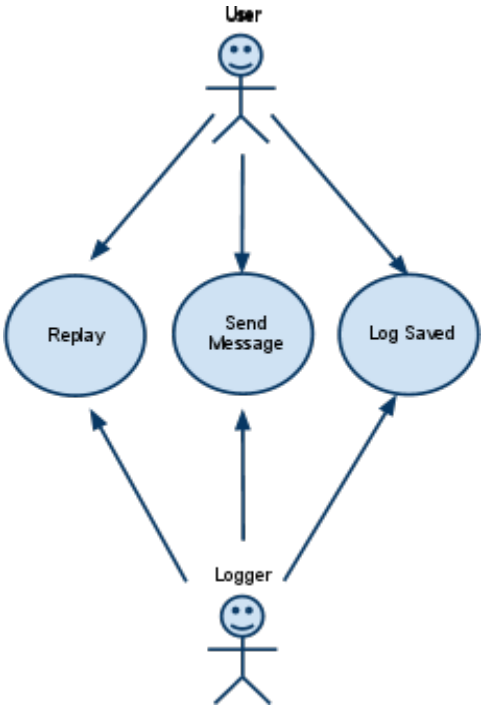


Fig. 3c: Use case diagram for logging.

3.1 Manual Simulation

3.1.1 Description and Priority

Manual simulation entails the manual creation and management of nodes before and during simulation. It also involves manual message passing. Actors include: User, Viewport, Node Manager, Simulator, Routing, and Logging. It is highly beneficial and relatively costly to implement.

3.1.2 Stimulus/Response Sequences

User actions are as follows:

1. Select simulation mode.
2. Add node.
3. Edit node position.
4. Edit node parameters.
    a. Add another node.
    b. Edit node position.
    c. Edit node parameters.
5. Queue a message.
    a. Queue another message.
6. Start simulation.
7. Add node.
8. Edit node position.
9. Edit node parameters.
10. Queue a message.
    a. Queue another message.
11. Stop simulation.
12. Save log to file.

3.1.3 Functional Requirements

Req #	Description
FR-1	The simulator shall contain a viewport that has the capability to display a graphical representation of a wireless network.
FR-1.1	The viewport shall provide the ability to accept mouse events for node selections.
FR-1.2	The viewport shall provide the ability to accept mouse events for node arrangements.
FR-2	The simulator shall provide input fields for entering network-related data and node information.
FR-3	The simulator shall provide input fields for entering node-related data.

FR-3.1	The input fields provided for entering node-related data are: ● Name ● IP Address ● X (position on the viewport's x-axis) ● Y (position on the viewport's y-axis) ● Power
FR-3.2	The simulator shall have the capability to validate newly entered node data.
FR-3	The simulator shall provide an input field for searching for a node.

FR-4	The simulator shall have the ability to display node information in a text area.
FR-4.1	The node information to be displayed are: ● Name ● Message
FR-5	The simulator shall have the ability to display network information in a text area.
FR-5.1	The network information to be displayed are: ● TBD
FR-5	The simulator shall provide an "Add" button that contains the ability to create new nodes.
FR-6	The simulator shall provide a "Delete button that contains the ability to remove existing nodes.
FR-7	The simulator shall provide the an options menu that contains the ability to modify routing protocols.
FR-7.1	The routing protocols are: ● AODV ● DSDV
FR-8	The simulator shall have the capability to log system errors in a text file.
FR-9	The simulator shall have the capability to catch and display system errors to the log.
FR-10	The simulator shall provide a "Replay" button that contains the ability to re-display the recorded log data from the previously successful simulation.
FR-11	The simulator shall provide a Help button that opens the User Guide for further assistance.

3.2 Automated Simulation

3.2.1 Description and Priority

An automated simulation entails the automated creation and management of nodes. It also involves

automated message passing according to some preset. Actors include: User, Viewport, Node Manager, Simulator, Routing, and Logging. It is highly beneficial and relatively costly to implement. It is based on the implementation for manual simulation.

3.2.2 Stimulus/Response Sequences

3.2.2.1 User Action and System Response

User actions are as follows:

1. Select simulation mode.
2. Load simulation preset.
3. Start simulation.
4. Stop simulation.
5. Save log to file.

3.2.3 Functional Requirements

The functional requirements are identical to those in the section for manual simulation.

3.3 Replays

3.3.1 Description and Priority

Initialization and execution of replays is a high priority feature responsible for the playback of previously recorded simulations. Actors include: User, Viewport, Node Manager, Simulator, and Logging. It is highly beneficial and relatively cheap to implement.

3.3.2 Stimulus/Response Sequences

User actions are as follows:

1. Select replay mode.
2. Load log file.
3. Press play.
4. Press pause.
5. Press play.
6. Exit replay mode.

3.3.3 Functional Requirements

Req #	Description
FR-1	The simulator shall provide a “Start” button that contains the ability to start the playback functionality at the simulator’s current time step.
FR-2	The simulator shall provide a “Stop” button that contains the ability to stop the playback and exit the replay functionality.
FR-3	The simulator shall provide a “Pause” button that contains the ability to pause the playback functionality at the simulator’s current time step.
FR-4	The simulator shall provide a “Restart” button that contains the ability to restart the playback functionality at the simulator’s initial time step.
FR-5	The simulator shall provide a “Speed Up” button that contains the ability to increase the replay functionalities speed.
FR-6	The simulator shall provide a “Slow Down” button that contains the ability to decrease the replay functionalities speed.
FR-7	The simulator shall provide a menu option that contains the ability to load and replay of the previous simulation.

4. External Interface Requirements

4.1 User Interfaces

The GUI interface will contain all controls needed to build and simulate an ad-hoc wireless network and will closely resemble the AODV Simulator [1]. The GUI interface will be constructed using Java Swing, which will contain a node manager, replay, and logging components. The interface will utilize common user-friendly tools and selection items such as a drag and drop function, drop down menus, pick lists, input fields and clearly defined buttons.

Where applicable hover tool tips will be added to provide additional clarification.

Error messages will be displayed in dialog boxes which give the option “OK” then will return to the main simulator pane.

The simulator’s user interface will provide two distinct modes. Each mode will give users the ability to perform specific functions. This is done to try to simplify the application by separating node construction from implementation.

4.1.1 Simulation Mode

Simulation Mode, see Figure 4.1.1 Simulation Mode Mock-up, allows the user to construct their wireless network. Users can easily do this by utilizing the tool bar’s Add Node and Delete Node drag and drop regions. This allows users to add as many nodes as they need just by dragging and dropping from the Add Node region of the tool bar. Users can then delete nodes as needed by selecting the desired node dragging and dropping the node onto the Delete Node region.

The Node Data region provides users with the ability to customize each node. Users can change the node's name, position on the viewing region, enter a message and select which nodes to attempt to send the message to.

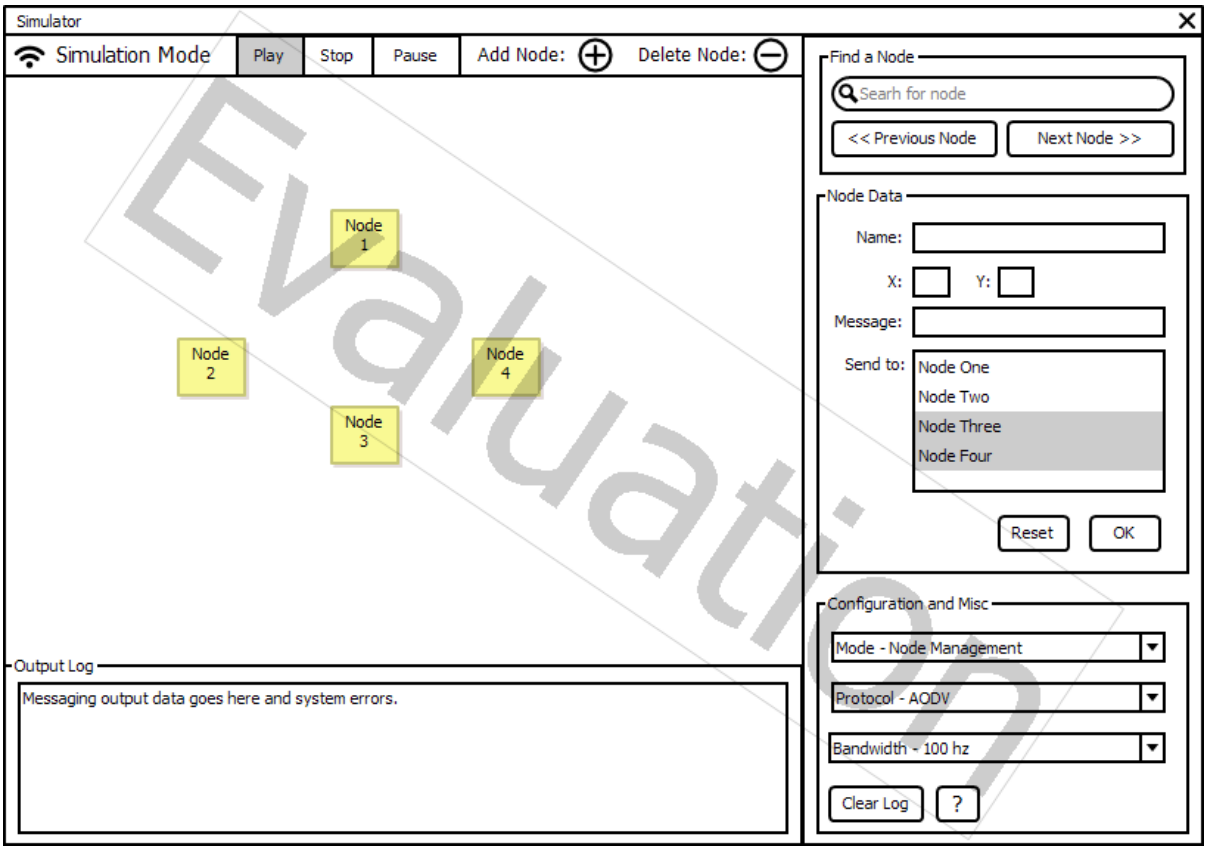


Fig 4.1.1 Simulation Mode Mock-up

4.1.2 Replay Mode

While in Replay Mode, see Figure 4.1.2 Replay Mode Mock-up, the Simulation Mode specific functions are either not visible or disabled. The Node Data region is available, but disabled, so users can still view each nodes data. This isolates can changes to the simulation and allows users to focus on the current simulation.

The Replay Mode will contain a Speed option, that allows users to slow down or speed up the replay of the previous simulation.

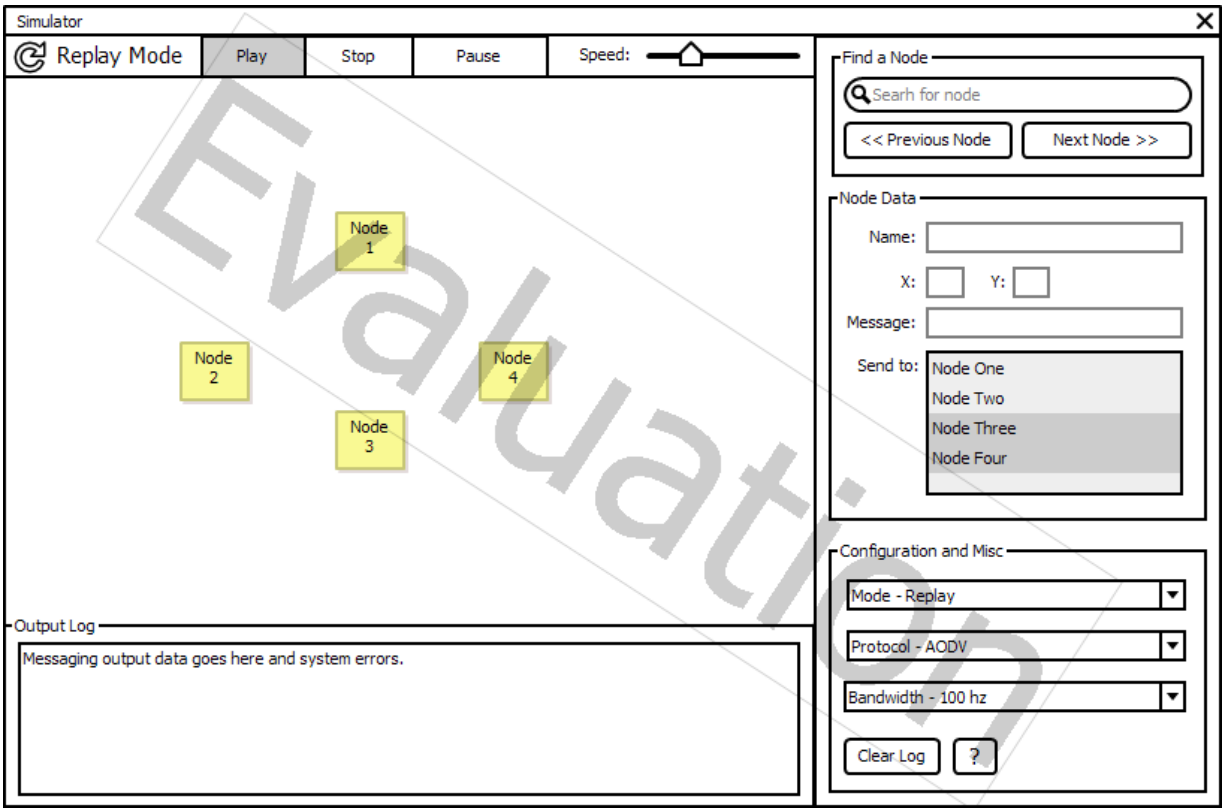


Fig 4.1.2 Replay Mode Mock-up

4.1.3 Common User Interface Functions

There are common functionalities that are available and active across both modes. One of these functionalities is Find a Node. This functionality allows users to search for a certain node by entering search criteria, such as a node's name. If found, the node is selected on the viewing region and the node's data is populated in the Node Data fields. Also, within the the Find a Node region, buttons are provided that will find either the previous node or the next node on the screen. The previous and next node buttons use a nodes created date timestamp to determine which node is next or previous. If no nodes are available, then the previous and next buttons are disabled. If there are no more previous or next nodes to iterate over, then that particular button is disabled.

Other common functions are contained within the Configuration and Misc region. This region allows users to select between the two modes, select between two messaging protocols and select from a variety of wireless bandwidths. Also, available to users is the Clear Log button that clears the Output Log displayed on the GUI interface. (Note: The text file log with not be affected by the Clear Log button) Finally, within this region a help button ("?") is provided that gives users access to documentation about the software.

In addition, the Output Log is another common functionality between the two modes. This functionality displays to the user what is going on in the background with the simulation is being run.

Also, both modes can Start, Stop or Pause the simulation run or replay.

4.2 Hardware Interfaces

All interfaces with user hardware will occur through the existing platform (operating system) and will be handled by the JVM. The interfaces that will be required by this software will include a display (monitor) output, keyboard input, and mouse (point and click) input. Maximum screen resolution may be a useful piece of information as it could limit the size of the simulation window. Our software will never access these components directly, this will be done using existing Java libraries designed for this purpose.

4.3 Software Interfaces

As with hardware the JVM will handle any interactions between the simulator and the existing software. The possibilities of interaction are limited only by the capabilities of Java.

4.4 Communications Interfaces

Aside from the aforementioned required hardware and software interfaces (monitor, keyboard, mouse) the software will only communicate with the system on which it is running. Those interactions will be limited to the ability to output data to a file on the local system and possibly to interface with the system clock in order to record timestamps on events (if so required). These functions will also be handled by the Java platform.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Generally, we want the GUI to be as responsive as possible. Actions associated with mouse and keyboard events should be responded to immediately by the system.

With regard to simulation speed, it is important to make sure that it is not too fast. Each step in the simulation should be comprehensible to humans. At the same time, a simulation speed that is too slow might irritate the end user.

We also want to make sure that our node search feature is as fast as possible.

5.2 Safety Requirements

As this simulator is purely virtual, there are few, if any, losses or damages that could result from the use of this product. With regard to safety, there are no safeguards or actions that must be taken, and there

are no actions that must be prevented. Furthermore, there are no safety-related external policies or regulations that affect the product, and there are no safety certifications that must be satisfied.

5.3 Security Requirements

Since the product is a simulator, not a real wireless network, there are no concerns regarding security or privacy. There are no user identity authentication requirements, no external policies or regulations regarding security issues, and no certifications that must be satisfied. If a user maintains the software to avoid any vulnerabilities inherent to the Java version used the security posture of the system should not be compromised.

5.4 Software Quality Attributes

The following software quality attributes are listed in order of decreasing importance:

- **Correctness** - The routing protocols must be correct because the system revolves around the accurate simulation of message passing. Furthermore, the output of the simulation is expected to be correct so that simulation logs can be exchanged between groups working on the same project.
- **Usability** - We want the GUI to be intuitive and easy to use. We are prioritizing ease to use before ease of learning. We want it to be easy for new users to start running simulations. Then, if they are interested, they can learn more about other features of the program.
- **Re-usability** - Our system should have reusable components. In particular, the simulation logging module should be reusable.
- **Maintainability** - We want to follow good design principles in creating our software so that future changes are as easy to implement as possible.
- **Adaptability** - We want our system to be easily extended so that, for example, it is easy to add new routing protocols.
- **Robustness** - Our system should have good exception handling and should fail gracefully.
- **Reliability** - Our software should perform consistently and fail as infrequently as possible.
- **Testability** - Our system should be unit testable so that it is easy to check for bugs in each module.
- **Portability** - We want our system to be cross-platform. This should be simple because Java is cross-platform.
- **Availability** - Our system will be publicly available under the GNU GPLv3 license, so it will be free for educational use.
- **Flexibility** - The simulator will allow configurations to as flexible as the system will allow without hindering usability.
- **Interoperability** - The simulator will be developed in a modular fashion. This allows other systems to more easily take and use specific sections of the source code for their our educational purposes.

6. Other Requirements

This software will be developed concurrently with other parties developing a similar application with the same requirements. A requirement has been given that one component of this system be interchangeable with that of the other two groups. As assigned this function is the “simulation logging” component. This component will log events that occur during the simulation run. A common data-structure which includes [Time Stamp|Event|Event Class|Message Size|IP Src|IP Dest|Intermediate|Hops|Protocol] at minimum will be necessary.

A suggestion was made to handle this data-structure as a vector and to dump the logging data straight to a text file, however additional clarification will be required. Close collaboration with the other developers on this component will be required to assure maximum interchangeability and to avoid extraneous variables and the need for software adapters.

Appendix A: Glossary

AODV - Ad hoc on-demand distance vector is a reactive routing protocol that only requests a route when it needs one. Nodes are not required to maintain routes to destinations that are not communicating.

CONOPS - Concept of Operations document is a user-oriented document that describes system characteristics for a proposed system from the users' viewpoint.

DSDV - Destination-sequenced distance vector is a proactive hop-by-hop distance vector routing protocol where each network node maintains a routing table that contains the next hop to any reachable destination as well as the number of hops required to them. The routing table is updated via periodical broadcasts.

GUI - Graphical User Interface; an interface that allows a user to interact with a program through panels, buttons and other means not solely limited to text.

J2EE - Java Platform, Enterprise Edition or Java EE is a widely used platform for server programming in the Java programming language.

JRE - Java Runtime Environment is the execution platform for the JVM.

JVM - Java Virtual Machine is the platform which allows Java bytecode to be run (through the JRE) on multiple platforms

Lines of Code (LOC) - The number of lines of code used in a piece of software.

Mobile Ad Hoc Network (MANET) - A network formed without any central administration which consists of mobile nodes that use a wireless interface to send packet data.

Node - A connection point on a network.

Appendix B: Analysis Models

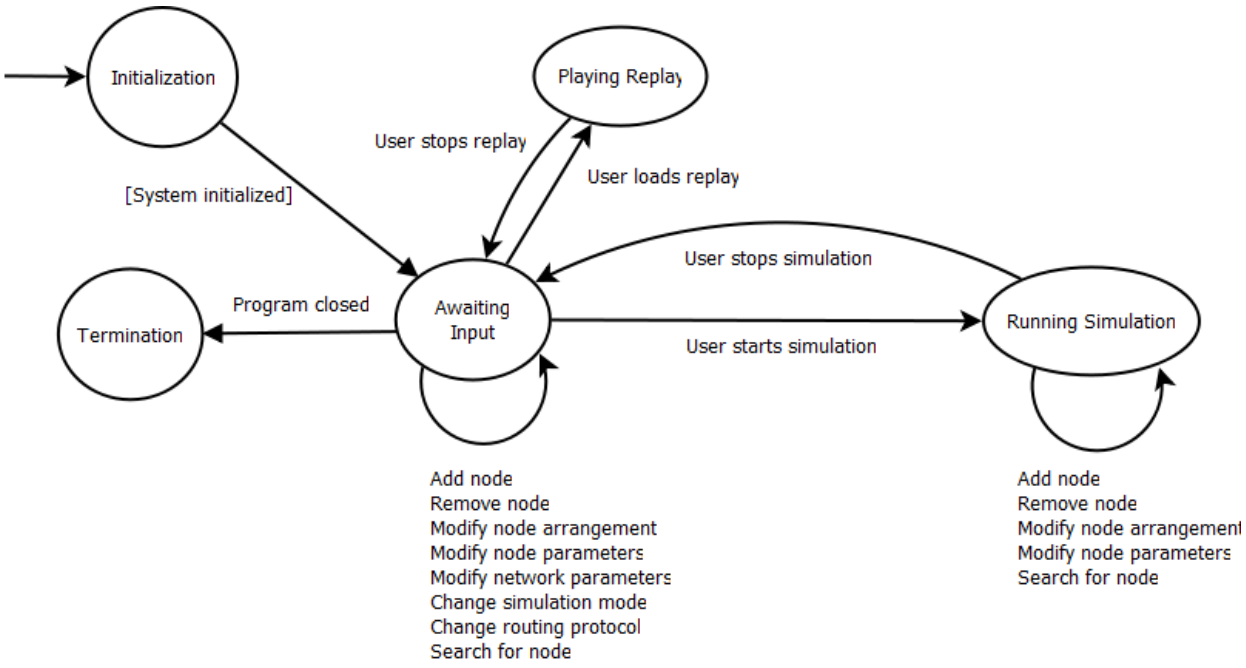


Fig. B: High-level state-transition diagram

Appendix C: Issues List

Pending decisions:

- Further information and clarification on the shared component.

Information that is needed:

- Implementation details for replays.

Conflicts awaiting resolution:

- Design of modular simulation logging system.