A Visualization and Animation Tool for NS-2 Wireless Simulations: iNSpect*

Stuart Kurkowski, Tracy Camp, and Michael Colagrosso
Dept. of Mathematical and Computer Sciences
Colorado School of Mines
1500 Illinois Street
Golden, CO 80401
{skurkows,tcamp,mcolagro}@mines.edu
Corresponding Author: Stuart Kurkowski
Phone: 303-648-9004 Fax: 303-663-1757

June 16, 2004

Abstract

The Network Simulator 2 (NS-2) is a popular and powerful simulation environment, and the number of NS-2 users has increased greatly in recent years. Although it was originally designed for wired networks, NS-2 has been extended to work with wireless networks, including wireless LANs, mobile ad hoc networks, and sensor networks. The Network Animator (NAM) for NS-2 has not been extended for wireless visualization, however. In this paper, we discuss a new visualization and animation tool for use with NS-2 wireless simulations. Visual analysis of a wireless environment is important for three areas of NS-2 based simulation research: (1) validating the accuracy of a mobility model's output and/or the node topology files used to drive the simulation; (2) validation of new versions of the NS-2 simulator itself; and (3) analysis of the resulting NS-2 trace files. Our iNSpect program can handle all three of these areas quickly and accurately. We've made our iNSpect program available for other researchers in order to improve the accuracy of their simulations. See `http://toilers.mines.edu` for information on obtaining iNSpect.

I. INTRODUCTION

The number of wireless network devices will soon surpass the number of wired devices [7], and research in the area of wireless networking is increasing at a similar rate. There are many wireless

*This work supported in part by NSF Grant ANI-0208352. Research Group's URL is *http://toilers.mines.edu.*

network simulators and the Network Simulator-2 (NS-2) [22] with the Monarch project wireless extension [21] is popular among wireless researchers. NS-2 was initially a wired simulator developed from the REAL network simulator [22]; the NS-2 effort is supported by the VINT Group and NSF.

A visualization tool is needed to understand the large amount of data produced during network simulations, and NAM (Network Animator) is the default NS-2 animator. The human visual system is unrivaled in pattern recognition and offers the ability to process large amounts of data quickly and clearly [1]. Visualizations add to the understanding gained via statistical analysis. For these reasons, NAM was designed to provide a graphical user interface for the creation of wired network topologies [15]. It has an extensive environment for wired network development as well as trace file playback. Playback for the wired environment includes the display of links and packet flows.

NAM has not been extended under the Monarch effort to visualize wireless networking. There was an effort in the late 1990s to develop Ad-hockey [20] as a visualization tool for NS-2 wireless, but that has not continued. Currently, NAM displays only the node positions and movement in the network through the simulation window. With the increased demand for NS-2 simulations for wireless networks, a robust visualization tool is needed for wireless networks. There have been more than a half-dozen emails on the NS-users mailing list in recent months asking for video or animation support for wireless networks in NS-2 (see [13] and [6], for example). The increasing complexity of node and protocol behavior is driving the need for a visualization tool.

A visualization tool aids at least three specific areas of wireless simulation based research. First, the display of the node topology is necessary to understand the results of the simulation. To validate the simulation inputs, e.g., the mobility model or a manually developed topology, a visualization tool is needed. A visualization tool makes it possible to see the new generation of more realistic mobility models and their complex node behaviors. Second, new versions of NS-2 appear regularly and researchers are responsible for verifying their simulations are valid with the model implemented in NS-2 [22]. Third, after developing new protocols and techniques the results of the NS-2 simulation (in the form of trace files) must be analyzed both statistically and visually to ensure they match the research hypothesis and analysis.

We introduce our *interactive NS-2 protocol and environment confirmation tool* (iNSpect). The iNSpect program allows the visualization and animation of NS-2 based wireless simulations. Because it can animate a mobile ad hoc network without running NS-2 itself (by reading the mobility file, which is an input to NS-2) and because it can post-process successful NS-2 simulations (by reading the trace file, which is an output from NS-2), iNSpect is an agile tool that can be utilized with minimal overhead.

In this paper, we first provide background information on NAM's capabilities and then discuss our iNSpect tool. We address topology validation, model validation, and simulation results analysis using iNSpect. We then discuss a few additional areas of use for iNSpect, e.g., generating graphics for presenting simulation results and dynamic display of statistics. Information on obtaining our iNSpect program is available at `http://toilers.mines.edu`. To see iNSpect in action, go to `http://toilers.mines.edu/iNSpect`.

II. NAM AND iNSpect FEATURES

A. NAM

The Network Animator (NAM) is a visual tool that allows users to create Tcl based NS-2 scripts through a graphical user interface [15]. NS-2 wired simulation scripts have five basic elements: nodes, links, queues, packets, and agents. Nodes are the network devices that send and receive packets. Links are connections between nodes (explicitly defined for wired networks). Queues are the send and receive buffers on each node. Packets are the messages being sent between nodes via the queues and links. Finally, the agent is the protocol running on the node that determines the nodes' behavior for packet sending and receiving. These five elements are the building blocks for NAM wired animations. NAM visually depicts the nodes by trace file defined shapes. NAM represents wired links as lines between the connected nodes. NAM represents wired packets as blocks moving along the link corresponding to the route. During playback, packet and agent information is displayed in NAM monitor windows. The packet monitor shows the size, identification number, and time the packet was sent during the transmission period. Once a packet reaches its destination, it is removed from the monitor [18]. The agent monitor displays the name of the agent and the values of any trace variables that are set.

For node topologies, NAM supports three layout methods. Two (link orientation and automatic graph layout) are for wired networks where links are known, oriented, and permanent. The nodes can be placed based on the orientation of the links that pair nodes, or they can be automatically generated and allowed to converge over time [18]. The third method is for wireless networks, and it is an (x,y) coordinate layout method. This method is designed to animate a wireless node located anywhere in the Cartesian world. The addition of this third method is the only wireless unique functionality in NAM [18].

By adding Cartesian support to NAM, it can playback NS-2 generated trace files for wireless simulations. However, playback in the animation screen is limited to node movements and nodes emit a circular pattern to represent their transmission signal. The concepts of links and queues are not supported in the NAM wireless animations, because links and queues represent a permanent relationship between two nodes which do not exist in mobile wireless networks. In other words, NAM does not show the wireless links. Packet and agent monitors in wireless networks provide the same information as in the wired networks.

In summary, NAM is not a visualization tool designed for wireless analysis and validation. NAM was originally built for wired networks. While NAM can be used to show nodes moving, NAM does not show wireless links [15]. In other words, there is no visualization of transmission and routes generated when using NAM for NS-2 wireless scenarios.

B. iNSpect

The interactive NS-2 protocol and environment confirmation tool (iNSpect) is a C++ OpenGL-based [25] visualization tool that allows animation of wireless networks simulated in NS-2. The iNSpect program reads in NS-2 mobility files and the plays back simulation results. The iNSpect program produces a visual display of the nodes in a wireless scenario based on Cartesian (x,y)

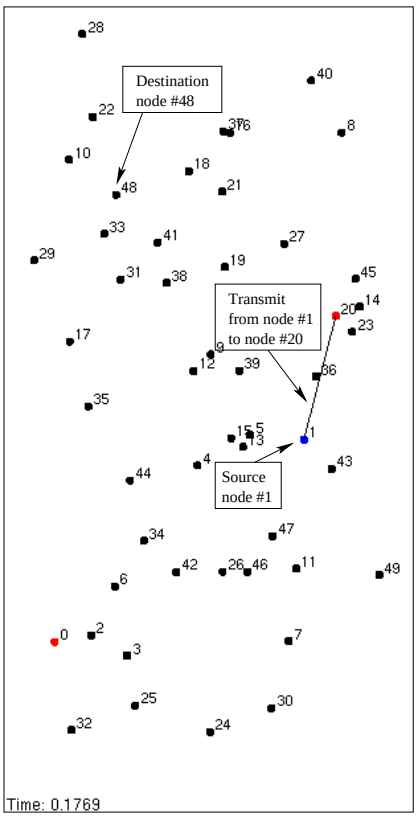

Figure 1: iNSpect showing the first hop of node 1 transmitting to node 48.

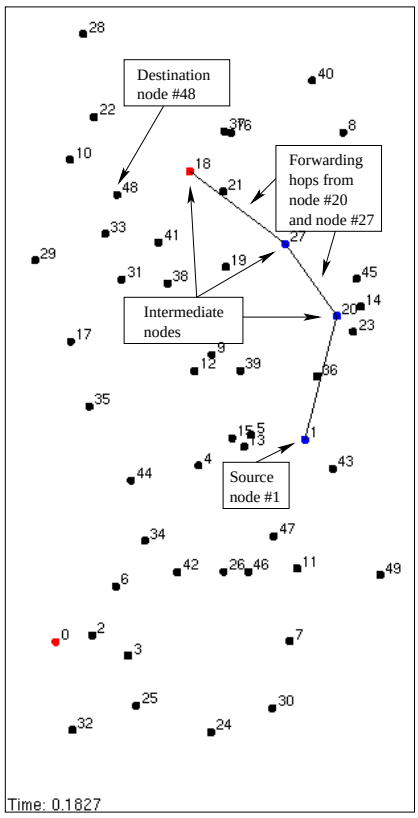

Figure 2: iNSpect showing the forwarding of the packet from node 20 through nodes 27 and 21 to node 18.

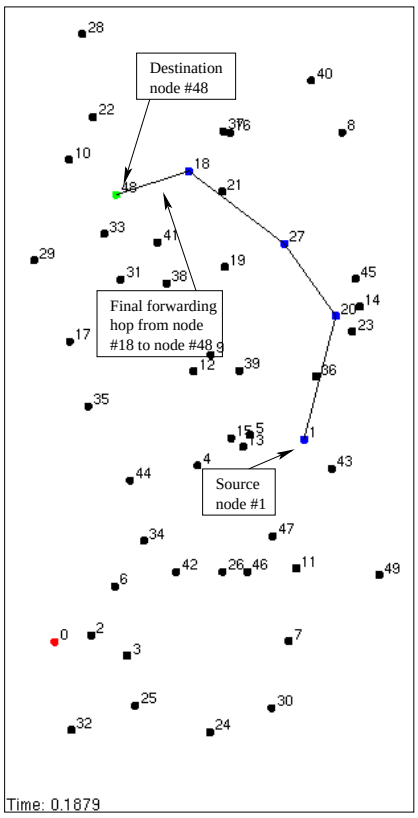

Figure 3: iNSpect showing the final hop of the route from node 1 to destination node 48.

Note: 50 nodes, 300 m x 600 m simulation area with 100 m transmission range.

coordinates used by NS-2. The iNSpect program has an event builder that reads a mobility file directly and schedules the movements and pauses of each individual node. The event builder functionality allows iNSpect to be used directly with mobility files generated by external mobility models. There is no requirement to generate a trace file from NS-2. Mobility file analysis outside of NS-2 makes iNSpect a must for mobility file validation, eliminating the overhead of additional lengthy executions of NS-2.

If and when a trace file has been generated, iNSpect uses the same event builder to read the trace file and schedule the packet transmissions with the node movements. Unlike NAM, iNSpect shows the wireless routes and the success or failure of a packet transmission. The transmissions are displayed with route lines and color coded nodes. When a node is transmitting to another node, a line is drawn between the two nodes. The line represents the attempt to transmit between the two nodes, similar to the link object in the NAM wired scenarios. The initiating node of a transmission attempt turns blue to indicate it is sending. The receiving node turns red until the packet is successfully received, then it turns green or blue. If the receiving node is the packet's final

destination it turns green. If the receiving node is not the final destination, it turns blue to show the next forwarding transmission attempt, and the route line is extended to show the forwarding of the packet. Figures 1-3 illustrate the three stages of a packet transmission from node 1 through nodes 20, 27, 21, and 18, to node 48. The persistence of the lines and node status allows for individual route analysis.

In Figure 1, node 1 initiates its packet send to node 48, by attempting to send the packet to node 20. In iNSpect the transmission is depicted as a line between node 1 and node 20. Node 1 turns blue and node 20 turns red until it successfully receives the packet. When node 20 receives the packet it turns blue because it successfully received the packet, but is not the final destination. Node 20 forwards the packet to node 27, which forwards the packet to node 18 (Figure 2). The iNSpect program continues to display the lines and node colors as the packet progresses to the destination. Node 18 then forwards the packet on its last hop to node 48 (Figure 3). Figure 3 shows the path from node 1 to node 48; node 48 turns green because it is the destination for the packet. The blue nodes along a path leading to a green node indicate a successful transmission to a destination, while blue nodes along a path leading to a red node shows failure of the packet to reach the destination and at which hop the packet failed. The graphical representation of the network activity gives the researcher more clues about individual success and failures of packets than the overall delivery ratio printed at the end of a scenario.

In iNSpect, the interaction between the node events and the display is one of stepping through the simulation time while calling for each node to update its location and activities. Because the playback is schedule-driven, iNSpect offers a variable playback speed as well as a variable start time for the simulation. That is, a researcher can jump to a particular point in a lengthy simulation trace file for detailed analysis.

The iNSpect program allows a user to display overlay objects such as circles, rectangles, and the current (x,y) location of each node. The geometric overlays may identify regions of interest. For example, Figure 4 illustrates a circular overlay at the center of the simulation area (150 m, 300 m) with a radius of 30 m. We use this circular overlay with a new mobility model that implements congestive movement for nodes in a given area of the simulation. In this new mobility model, a node moves according to the Random Waypoint Mobility Model [11, 3]. (The nodes are initialized in the steady state distribution of the Random Waypoint Mobility Model [17, 16].) Then as a node moves into the area of congestion, the node slows down. We use this circular overlay in iNSpect to represent the congested area, validating that the nodes slow in that area. This area can represent a food court at a mall or a large intersection in a city. The location and size of the circle is configurable within iNSpect.

A rectangular overlay is also provided in iNSpect. We use the rectangular overlays in evaluating Geocast routing protocols. In Geocast routing, packets are forwarded to nodes in a specific geographical area of the simulation [10]. For example, a city dispatcher may need to send emergency information to a certain area of a town to alert citizens of an evacuation. Figure 5 depicts a rectangular area of interest with corners at (150 m, 400 m) and (300 m, 600 m). The explicit representation of the rectangle on the iNSpect display allows visual analysis of a packet's route to the area.

The geometric overlays of iNSpect can be used to represent obstacles as well. As stated in

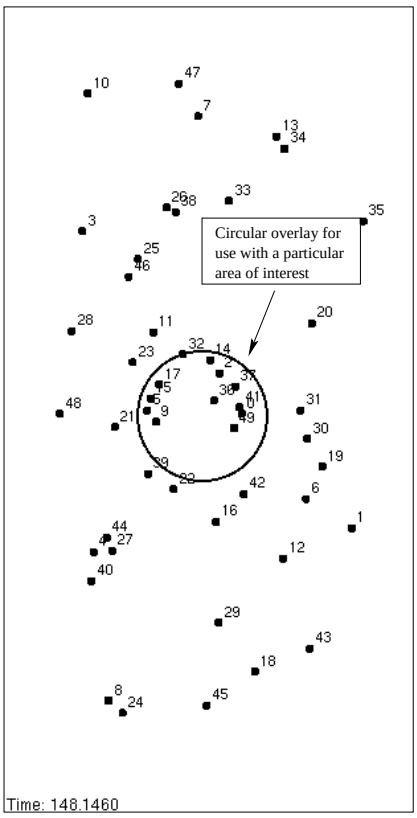

Figure 4: iNSpect display of a 30 m radius circle overlay in the center of the simulation area.

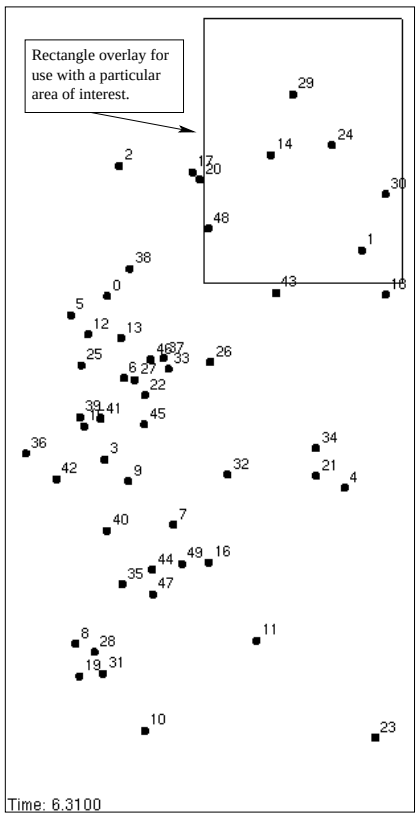

Figure 5: iNSpect display of a 100 m x 200 m rectangle overlay in corner of the simulation area.

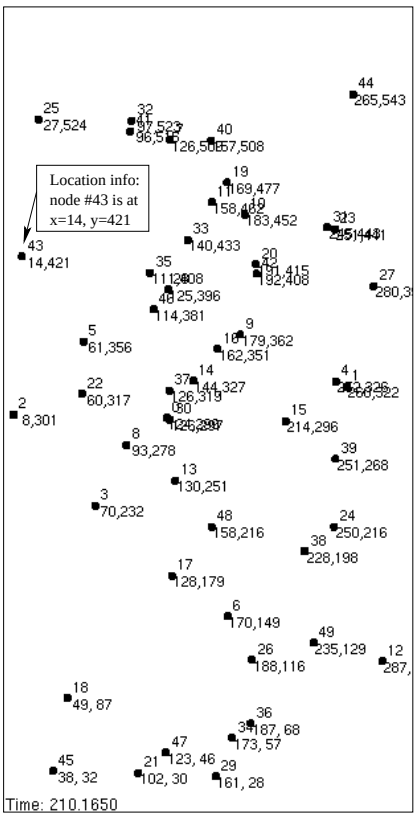

Figure 6: iNSpect display of the (x,y) coordinates for each node's location.

Note: 50 nodes, 300 m x 600 m simulation area.

[12] obstacles make mobility models more realistic. The obstacles affect both transmission and movement of nodes. The iNSpect program can be used to observe the affects of the obstacles on the movement of nodes and the transmission of packets.

The coordinate overlay, Figure 6, displays node position (x,y) coordinates. The location information can be used to evaluate accuracy of node location and movement against the researcher's plan for a node. Location information can also be used to evaluate location-based routing protocols. In location-based protocols, a node's knowledge of a destination's location is used to determine a route to the destination [14]. Using persistent routes and the node locations on the iNSpect display, a researcher can evaluate a protocol's performance on individual routes. This gives the researcher detailed information not available in summarizing statistics.

Our iNSpect program is driven by a configuration file, which minimizes the command line arguments and the need to recompile while enabling the user to control several aspects of the display. Activity persistence, and the position and size of overlay objects, are all part of the configuration file. The iNSpect program can display and animate the results of a single stand-alone mobility file,

it can animate an NS-2 trace file for movement, and it can animate a mobility file and an iNSpect unique trace file to show link activity. The iNSpect trace file is used to identify the wireless links not present in the NAM trace files.

III. iNSpect USES AND RESULTS

A. Topology analysis and validation

NS-2 uses node locations and speeds as an input to the simulator. This node information can be placed directly into the Tcl script or provided in an external file. A researcher needs to understand what node behavior NS-2 is calculating based on the location and movement information. For simple node arrangements, for example, one with only a few nodes and no movement, a diagram can be drawn by hand to depict the scenario. However, as we seek more realistic models to depict real-world movement [3], manual diagrams are (almost) impossible to generate. Many researchers currently base node location and movement on the mobility model generated mobility files [2, 23, 26]. These mobility models use mathematical algorithms and statistics to randomly or selectively position nodes and assign speeds throughout the simulation area. As the number of nodes in a simulation increases, and the number of variables such as location, speed, and pause time continue to increase, it is impossible to know what these mobility models are producing without a visualization.

Currently, the statistical way to validate a topology or mobility file for NS-2 is to analyze the individual lines in the mobility file, searching for certain characteristics such as node locations, etc. However, if the nodes have movement, this analysis is limited. Since NS-2 calculates the node movements during the simulation, the mobility file only contains initial node locations and direction or speed changes. Where the nodes are between these entries is calculated by NS-2 during the simulation and are not in the file. In other words, the complete analysis of a mobility file can only be done by running a simulation.

Until the development of iNSpect, the only way to visually validate a topology or mobility file was to generate the mobility file, run it through NS-2 to produce a NAM trace file, and then use NAM to analyze the mobility file for errors. This process can take several minutes or more, depending on the NS-2 simulator and the machine running the simulation. Additionally, the NAM trace file can grow to be on the order of hundreds of megabytes. These steps are tedious and wasteful for a developer of a mobility model, especially if there are many iterations.

Unlike NAM, iNSpect can process an NS-2 based mobility file directly. The iNSpect engine calculates the node movements from the mobility file just as NS-2 does in a simulation. Each update of the iNSpect display updates the node movements based on the playback speed. This capability streamlines the development of individual topologies or mobility files generated from an automated script or mobility model, because the nodes can be displayed and animated outside of NS-2. Not burdened with the many other things NS-2 does in a simulation, iNSpect can produce visual validation instantly for a given mobility file. Thus, quicker results for the model developer are obtained, and no trace files are written to disk. The direct processing of the mobility file allows a mobility model developer to complete many iterations quickly.

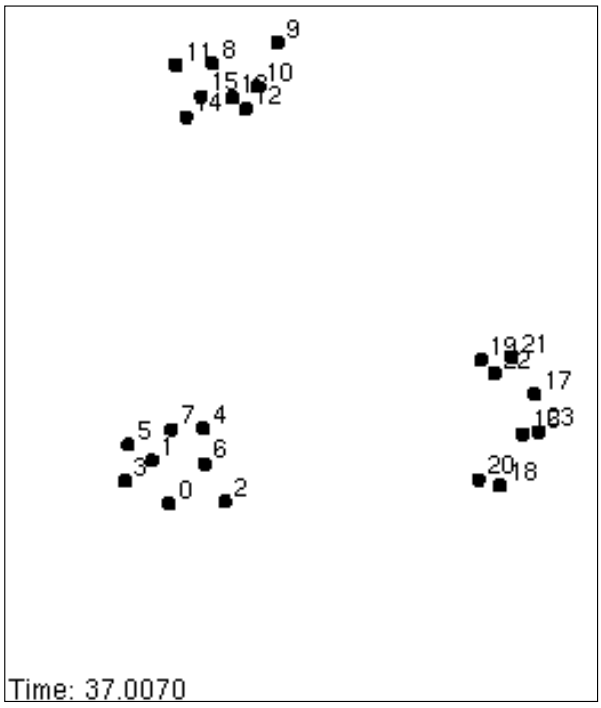

Figure 7: iNSpect displaying an RPGM model example with loosened travel from reference point.

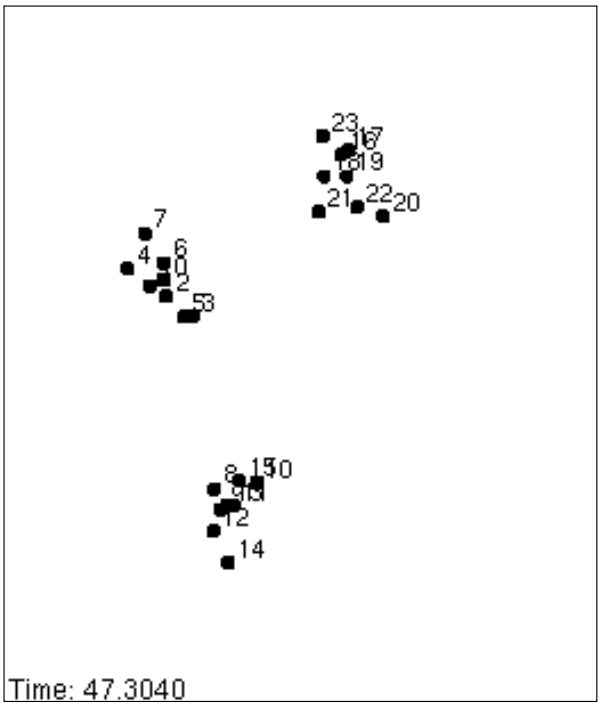

Figure 8: iNSpect displaying an RPGM model example with restricted travel from reference point.

Note: Simulation area is 600 m x 600 m with 3 groups of 8 nodes each

As an example of the animation process for iNSpect, consider the Reference Point Group Mobility (RPGM) Model [19, 3]. The RPGM model is used by researchers to represent rescue or search parties that move loosely as a group around certain reference points. To generate a mobility file from the RPGM model, the user must determine numerous parameters ranging from the simulation area's (x,y) dimensions, to speed and pause time, to the number of groups and number of nodes per group. The model also has two parameters that deal with the reference point separation distances and individual node wanderings from the reference point [3]. Figures 7 and 8 illustrate how a user can analyze these last two parameters in iNSpect. Figures 7 and 8 show two mobility files with the same dimensions, speed, pause time, number of nodes, and number of groups. The only difference is the looseness of the nodes' distances from the reference point. Immediately, the effect of the change is seen. The iNSpect program is the only way to see the affect of these parameters on the simulation from a mobility file directly. Without iNSpect, a user would first need to run the full simulation scenario in NS-2 to visualize the mobility model parameters in NAM. The iNSpect program can also be used to explain these parameter value differences. The contrast shown in Figures 7 and 8 is a clear way to explain the impact of these parameters to others.

The iNSpect program can do more than test parameters of existing models, because iNSpect allows the immediate validation of the files produced by a mobility model. That is, iNSpect can be used to develop new models. For example, we used iNSpect during the development of a new

congestion based mobility model. In this new mobility model a node will slow down if its number of neighbors exceeds a threshold. We used iNSpect in two ways. First, iNSpect gave us an instant look at the model results and allowed us to visually see the nodes slow down in congested areas. Second, iNSpect enabled us to discover a problem with the model. Specifically, we generated 2000 second scenarios with 50 nodes and no pause time. When we used iNSpect to animate the model's output, we saw the nodes slowing in congested areas as expected. However, near the end of the 2000 seconds, a few nodes stopped moving. With iNSpect, our implementation problem was debugged and fixed quickly.

B. Simulation model analysis and validation

As stated at the beginning of the NS-2 documentation [22] *"users of NS-2 are responsible for verifying for themselves that their simulations are not invalidated by bugs."* The question is how does one ensure a simulation is correct? While there is no way to guarantee correctness, iNSpect can help.

As an example, we recently installed the new 2.27 version of NS-2 and, similar to several accounts on the NS-mailing list (e.g., [4]), we noticed a significant drop in the performance of our simulations (i.e., delivery ratio, end-to-end delay, and overhead). Using iNSpect, we discovered an error in the simulator. Specifically, NS-2.27 does not update the position of a node unless there is an event for that node. (The error was concurrently located by the author of [24].) The NS-2.27 error is shown in Figures 9 and 10. In Figure 9, the simulation area is 300 m x 600 m and in Figure 10, the simulation area is 600 m x 600 m. Each node's transmission range is 100 m in both figures. As shown, there are several nodes that successfully transmit a packet outside the 100 m range, e.g., (node 9 to node 34 in Figure 9), (node 16 to node 8 in Figure 9), and (node 0 to node 26 in Figure 10). For example, in Figure 10, the distance between node 0 and node 26 is 453.4757 meters, which is well over the 100 m transmission range. Figure 11 shows NS-2.27's incorrect view of node 0's location and why the transmission was successful in the trace file.

In summary, iNSpect quickly illustrated the inconsistencies of the simulation output under the new version of NS-2.27. We also note that NAM could not have shown this problem for two reasons. First, NAM's output is based on the NS-2 model, therefore the nodes shown in NAM would be in the locations seen by NS-2 (i.e., Figure 11). Second, NAM does not show the links and packet flows. Thus, even if the nodes were in a different location, an analyst would not have seen the extreme transmission distance.

C. Simulation results analysis

An entire simulation (using both the mobility file and iNSpect trace file) can be animated with iNSpect. NAM animates simulations for wired networks in NS-2, but is unable to animate network traffic for wireless networks. The iNSpect display shows each transmission, with lines between nodes for transmission attempts and color codes for the sending nodes, nodes that receive a transmission successfully, and nodes that do not receive a transmission successfully. The iNSpect display shows the virtual link in a transmission, instead of the transmission ring shown in NAM. The ring, although representative of an omni-directional wireless signal without obstacles, does

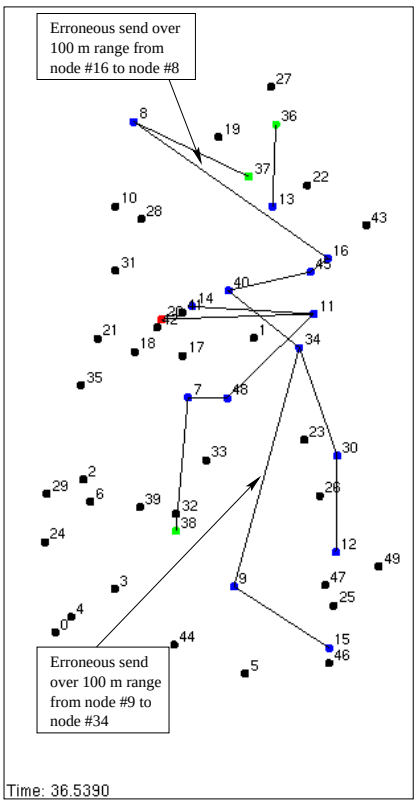

Figure 9: iNSpect showing an NS-2.27 model error. Two transmissions exceed the 100 m node transmission range. The simulation area is 300 m x 600 m.

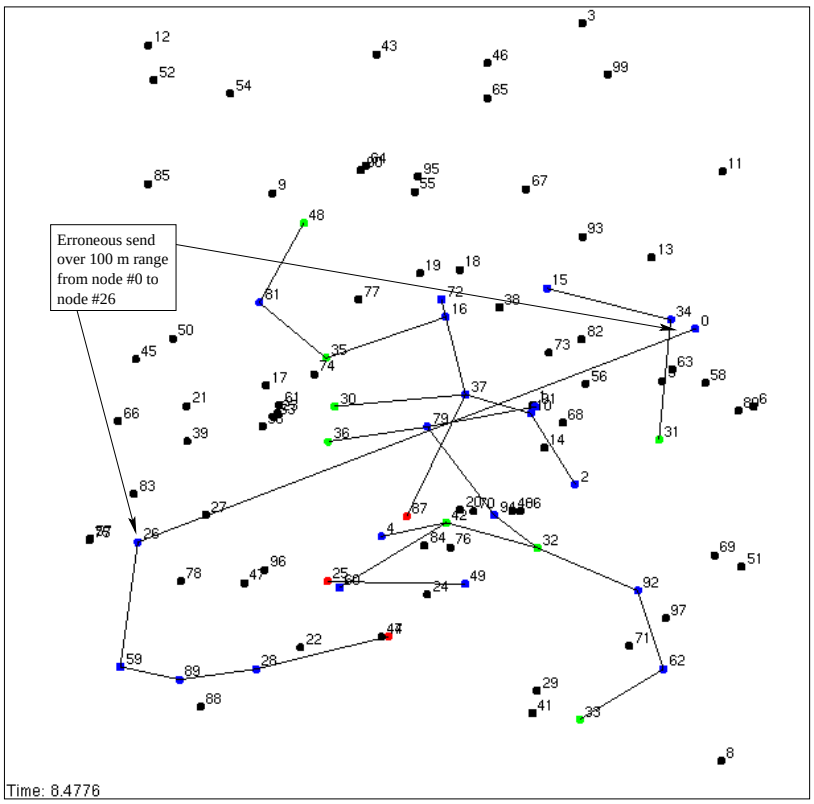

Figure 10: iNSpect showing an NS-2.27 model error. The node 0 transmission exceeds the 100 m transmission range. The simulation area is 600 m x 600 m.

not help a researcher trace the route of a packet. The iNSpect animation allows quick analysis of packet routes, as well as animation of issues. An animation of the results aids understanding of summary performance statistics such as delivery ratio, end-to-end-delay, and overhead.

By knowing the path a packet takes from source to destination, we can learn more about the behavior of a protocol. Figure 12 shows a snapshot of a Location Aided Routing (LAR) simulation [14]. LAR routing uses knowledge of the destination node's location to build routes for a packet transmission. We can use iNSpect to evaluate the number of hops a given successful transmission takes, and whether a protocol can be improved to reduce the number of hops. For example, in Figure 12 we see a successful transmission from node 5 to node 44. The path from node 5 goes through nodes 84, 22, 4, 101, 82, 45, 57, 11, 93, 64, 24, 77, and 44 (a total of thirteen hops). From the iNSpect program we have the time this event occurs and we see other more direct paths such as the one from node 101 to nodes 112 or 97, to 24. With this knowledge we can look at which routes were in the cache for node 5 and see why it did not discover this shorter route. Individual analysis like this would be impossible with just performance statistics and no iNSpect animation.

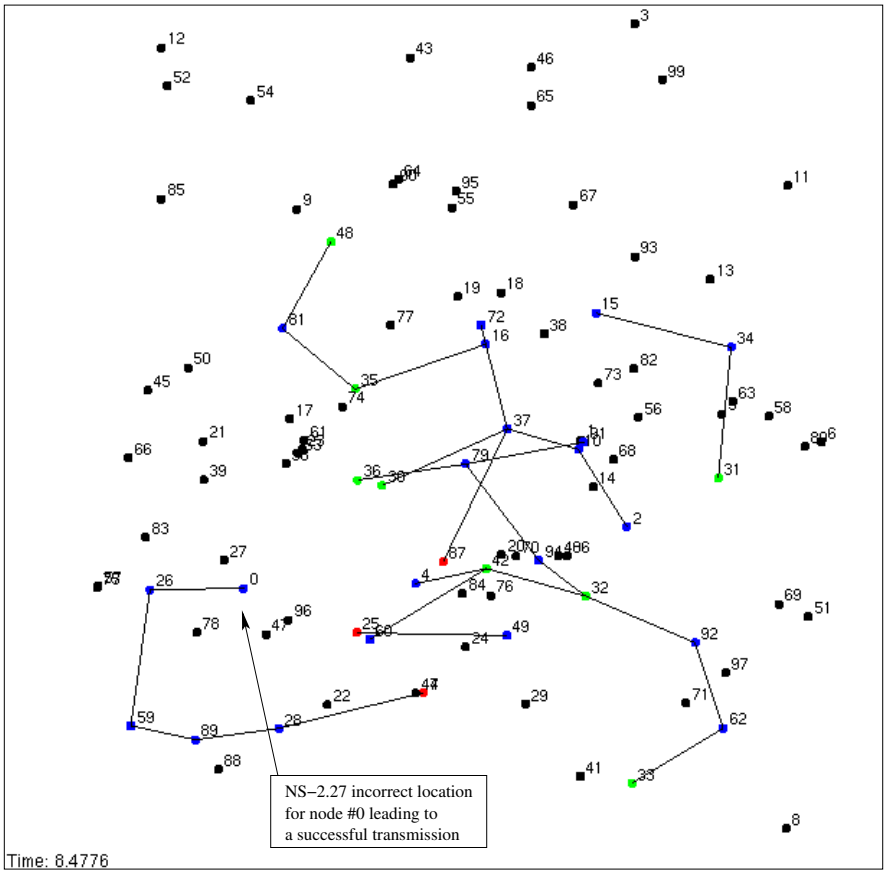

Figure 11: iNSpect showing NS-2's incorrect view of node 0's location at the same time as Figure 10. This incorrect view places node 0 in range of node 26, explaining the successful transmission.

We developed improvements to LAR that utilize the location information disseminated to find more direct routes [5]. Using our *projection method*, a node, A, sets its assessment delay (the time it waits before rebroadcasting a route request packet) proportional to the length of the projection of the vector from the sending node, S, to A ($\overrightarrow{SA}$) onto the vector from the source to the destination node, D ($\overrightarrow{SD}$). The longer the projection, the more direct the route is, and the shorter A will set its assessment delay. Figure 13 illustrates the results of our route optimization: A route with eight fewer hops for the same transmission (from node 5 to 76, 19, 92, 77, to 44) compared to the route shown in Figure 12. Visualizing our LAR projection method is a valuable step in designing, analyzing, and communicating our improvements to the protocol.

We did a similar analysis with iNSpect in an effort to implement the Zone Routing Protocol (ZRP) [9] in NS-2. In ZRP each node maintains detailed information about nodes within its zone. A node's zone size is defined by the number of hops from the node. For example, if a node had a zone size of three, it would keep track of all nodes within 3 hops. By keeping track of nodes inside a zone, a node has an immediate route to those nodes. If a node needs a route to a node outside its zone, it uses a reactive protocol known as the Inter-zone Routing Protocol (IERP) [8]. The IERP is

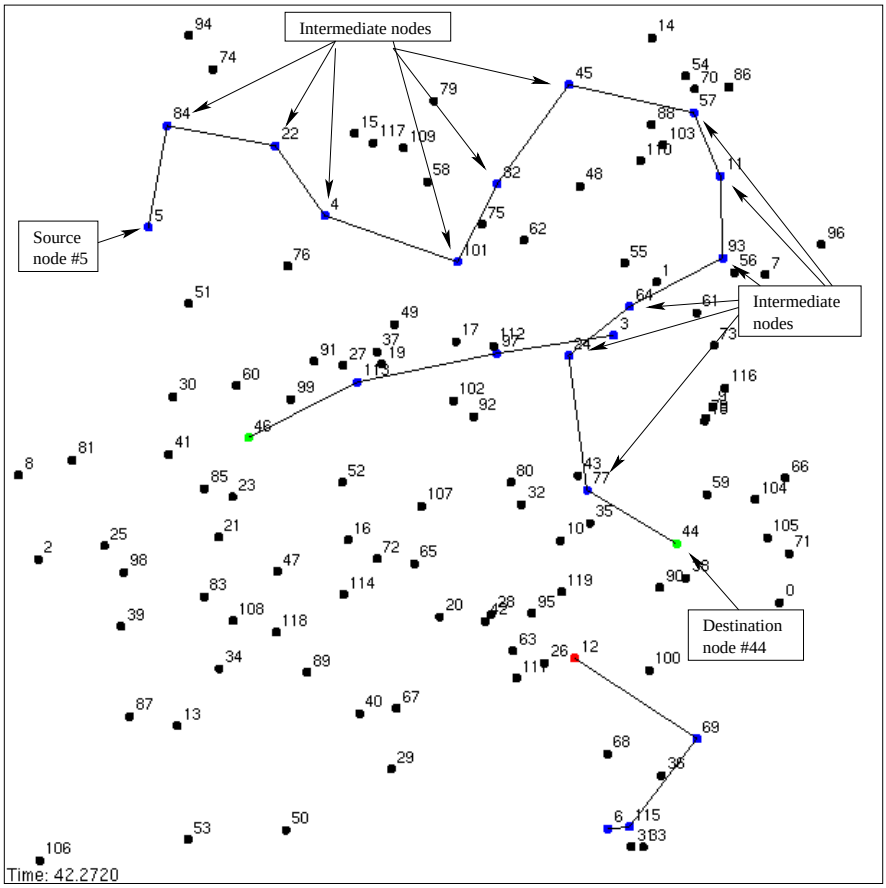

Figure 12: iNSpect showing a Location Aided Routing route selection for a transmission from node 5 to node 44 in a 600 m x 600 m simulation area with a 100 m transmission range.

used to request a route for this destination node. This is a costly request. Thus, when an Inter-zone route is returned, the nodes cache that information for possible future use.

By visualizing the ZRP results from NS-2 in iNSpect, we noticed that several of the nodes would only send packets a few hops beyond their zone and then the transmissions would fail. This was reflected by a low delivery ratio for each simulation, but there was no way to know the cause. Since we saw a trend of unsuccessful transmissions outside a node's zone, we adjusted the IERP cache timeout and ran the simulation again; the delivery ratio and the transmissions improved. In summary, iNSpect showed us that the routes outside the zone were bad, which led us to discover the IERP caches were maintaining stale routes. The iNSpect program gave us information that summary statistics could not.

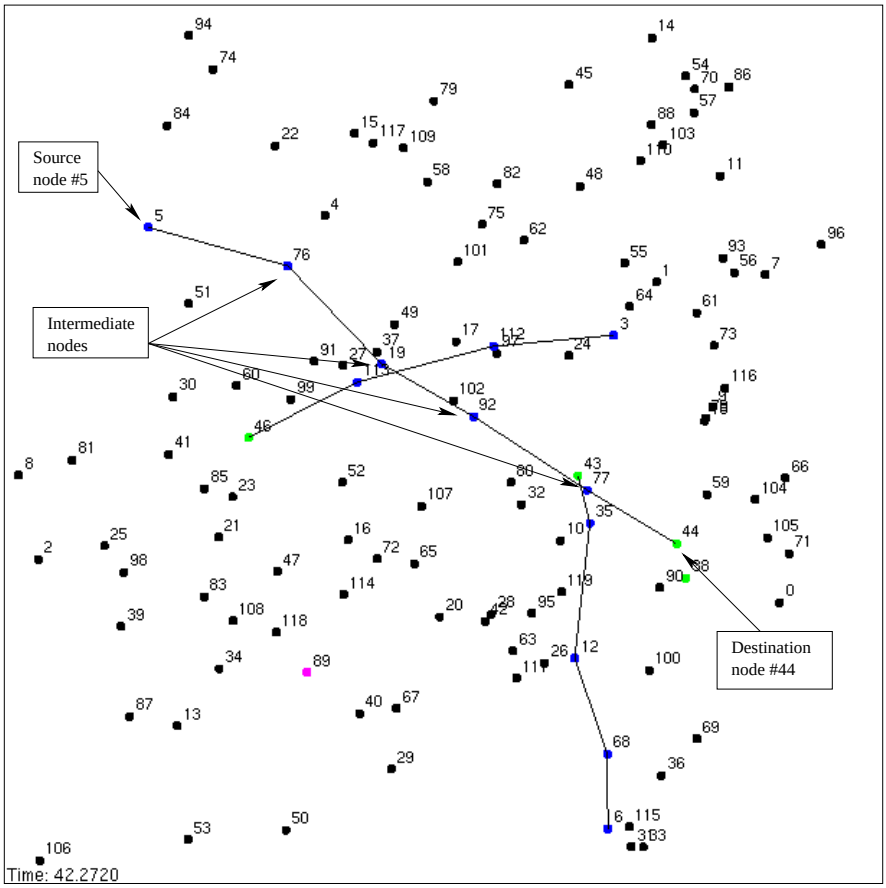

Figure 13: iNSpect showing our *projection method* [5] of Location Aided Routing route selection for a transmission from node 5 to node 44 in a 600 m x 600 m simulation area with a 100 m transmission range.

D. Additional uses

Finally, iNSpect can also be used to validate transmission or protocol behavior. For example, suppose nodes are placed at varying distances around a test node. The test node then attempts to transmit to each node. The resulting trace file and iNSpect can verify the communication successes of nodes within the transmission range and communication failures of nodes outside the transmission range.

The iNSpect program utilizes the OpenGL [25] libraries for rendering the animations. These libraries are the same libraries used in popular graphical games. The popularity has made many tools available for OpenGL based applications. The tools range from screen capture software to high quality movie making software. The iNSpect program is an excellent tool for making presentations of simulation results. The quick and clear animations of iNSpect make clean movies for displaying results. There are several tools for making animated GIF and full MPEG based movies of OpenGL applications.

Furthermore, because iNSpect is C++ and OpenGL code, it is easy to write front end processing units for it. The straightforward code can easily be extended to process different types of trace files, mobility files and events. The overlay patterns present in the current code can be extended to include other OpenGL-based rendering functions.

IV. FUTURE WORK

We are looking at expanding iNSpect in several areas. The first is the incorporation of more of the additional uses listed in section III D. For example, we plan to include the movie making capability directly into iNSpect for use with presentations. Also, we plan to make iNSpect work correctly with other NS-2 trace file formats and other simulator's trace files. Another area is including visual statistics in iNSpect. Visual statistics allow the different types of analysis to compliment each other in one tool. One idea to have iNSpect provide histograms and other charts about the node's positions. For example, a histogram of the volume of nodes at each x-coordinate and y-coordinate allows comparison with what the steady state model [17, 16] predicts for node position.

V. CONCLUSIONS

With the increase in wireless network research, visualization and animation of the node behavior, simulations, and results are a must. The iNSpect program is a quick low-overhead solution to animating mobility model files and topologies for NS-2. By using the iNSpect tool, a researcher can quickly discover anomalies in topology files, the NS-2 model itself, or even in the results of a particular protocol. In addition, because iNSpect runs outside of NS-2, it is fast and can save numerous hours of detailed detective work trying to validate results. Our iNSpect program can be used in many different ways to enable the quick and accurate inspection of wireless network scenarios with NS-2. As we have seen in our own research, iNSpect can reveal issues that summary statistics cannot. From analyzing node movement to packet routing, iNSpect can provide insight not available from totals and averages. Our tool is useful for simulations of large sensor networks, a simple wireless LAN, or a mobile ad hoc network. The iNSpect program supports wireless network capabilities of NS-2 and lets the human visual system participate in the analysis. Information on obtaining our iNSpect code is available at `http://toilers.mines.edu`. To see iNSpect in action, go to `http://toilers.mines.edu/iNSpect`.

VI. ACKNOWLEDGMENTS

We thank Dr. Jeff Boleng for the mobility file parsing code. We thank Ed Krohne for the inspiration to develop a tool to visualize a mobility file and a visualization trace file.

References

[1] E. Angel. *Interactive computer graphics: a top-down approach with OpenGL*. Addison Wesley, 1997.

[2] C. Bettstetter. Smooth is better than sharp: A random mobility model for simulation of wireless networks. In *Proceedings of the Fourth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'01)*, pages 19–27, 2001.

[3] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing (WCMC)*, 2(5):483–502, 2002.

[4] Jivei Chen. DSR performance is too bad in ns2, why? http://mailman.isi.edu/pipermail/ns-users/2004-March/040565.html. Page accessed on June 14, 2004.

[5] M. Colagrosso, N. Enochs, and T. Camp. Improvements to location-aided routing through directional count restrictions. In *Proceedings of the International Conference on Wireless Networking (ICWN)*, 2004. To appear.

[6] Jyoti Grover. How to draw graphs from ns trace files. http://mailman.isi.edu/pipermail/ns-users/2004-May/041965.html. Page accessed on June 14, 2004.

[7] A. Gurtov and S. Floyd. Modeling wireless links for transport protocols. Submitted to ACM Computer Communications Review, 2004.

[8] Z. Haas, M. Pearlman, and P. Samar. The interzone routing protocol (IERP) for ad hoc networks. Internet Draft: draft-ietf-manet-zone-ierp-02.txt, July 2002.

[9] Z. Haas, M. Pearlman, and P. Samar. The zone routing protocol (ZRP) for ad hoc networks. Internet Draft: draft-ietf-manet-zone-zrp-04.txt, July 2002.

[10] X. Jiang and T. Camp. A review of geocasting protocols for a mobile ad hoc network. In *Proceedings of the Grace Hopper Celebration (GHC 2002)*, 2002.

[11] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Routing protocols for mobile ad-hoc networks - a comparative performance analysis. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 195–206, 1999.

[12] E. M. Belding-Royer K.C. Almeroth, A. Jardosh and S. Suri. Towards realistic mobility models for mobie ad hoc networks. *(MOBICOM '03*, September 2003.

[13] Hou C. Kee. NAM support for wireless traffic. http://mailman.isi.edu/pipermail/ns-users/2004-May/042046.html. Page accessed on June 14, 2004.

[14] Y. Ko and N.H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'98)*, pages 66–75, 1998.

[15] J. Mehringer. The NAM editor. A presentation for the CONSER retreat, slide 2, 2001.

[16] W. Navidi and T. Camp. Stationary distributions for the random waypoint mobility model. *IEEE Transactions on Mobile Computing*, January-March 2004. To appear.See TR-1 for an early version of this paper.

[17] W. Navidi, T. Camp, and N. Bauer. Improving the accuracy of random waypoint simulations through steady-state initialization. In *Proceedings of the 15th International Conference on Modeling and Simulation (MS 2004)*, 2004.

[18] *The NS manual: formerly known as notes and documentation*, 2003. http://www.isi.edu/nsnam/ns/. Page accessed on June 1st, 2004.

[19] G. Pei, M. Gerla, X. Hong, and C. Chiang. A wireless hierarchical routing protocol with group mobility. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1538–1542, September 1999.

[20] CMU Monarch project. Ad-hockey. http://www.monarch.cs.rice.edu/cmu-ns.html. Page accessed on June 14, 2004.

[21] The Rice Monarch Project. The Rice Monarch extensions to the ns simulator. http://www.monarch.cs.rice.edu/cmu-ns.html. Page accessed on June 1, 2004.

[22] The VINT Project. The network simulator - ns-2. http://www.isi.edu/nsnam/ns/. Page accessed on June 1st, 2004.

[23] E. Royer, P.M. Melliar-Smith, and L. Moser. An analysis of the optimum node density for ad hoc mobile networks. In *Proceedings of the IEEE International Conference on Communications (ICC '01)*, pages 857–861, 2001.

[24] K. To. Bug in ns-2.27 wireless channel. http://mailman.isi.edu/pipermail/ns-users/2004-April/041388.html Page accessed on May 1st, 2004.

[25] M. Woo, J. Neider, and T. Davis. *OpenGL Programming Guide: The official guide to learning OpenGL*. Addison Wesley, 1997.

[26] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom '03)*, pages 1312–1321, 2003.