A Visualization and Analysis Tool for Wireless Simulations: iNSpect*

Stuart Kurkowski Tracy Camp Michael Colagrosso
Colorado School of Mines
skurkows, tcamp, nmushell, mcolagro@mines.edu

January 15, 2006

Abstract

Simulation is the tool of choice for researchers in the wireless ad hoc network community. As these simulations become more complex, tools are needed to analyze the large volume of output. In this paper, we discuss a new visualization and analysis tool for use with wireless simulations. Visual analysis of a wireless environment is important for at least three areas of simulation research: (1) visualization improves confidence in the accuracy of a mobility model's output and/or the node topology files used to drive the simulation; (2) visualization improves confidence in the correctness of new simulator versions themselves; and (3) visualization aids in analysis of simulation results. Our iNSpect program handles all three of these areas quickly and accurately. Our visualization tool, iNSpect, works directly with the NS-2 simulator; in addition, iNSpect can be used with any simulator or testbed output if the output is written in iNSpect's expected format. We have made our iNSpect program available for other researchers in order to improve the accuracy of their simulations.

1 Introduction

The number of wireless network devices will soon surpass the number of wired devices, and the amount of research in the area of wireless networking is increasing at a similar rate [9]. Wireless research often involves a testbed implementation and/or a simulation study. We conducted a survey [13] of MANET research published in the 2000-2005 proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) [7]. 114 out of the 151 MobiHoc papers published (75.5%) used simulation to test their research.

Network simulators allow researchers to analyze the behavior of these wireless devices at every level. As a result, these simulations are capable of producing very large amounts of data. The simulation community has made available many types of scripts (e.g., tracegraph [14]) to parse and analyze this output data, but visualization of the data is needed to further aid understanding of the output. A good visualization package is important, because the human visual system is unrivaled in pattern recognition and offers the ability to process large amounts of data quickly and clearly [3]. Visualization adds to the understanding gained via statistical analysis. As we show in this paper, certain erroneous network behaviors could go undetected without visualizations.

There have been many emails on the NS-users mailing list asking for visualization or video support for wireless networks in NS-2 (see [2] and [11], for examples). The increasing complexity of node and protocol behavior is driving the need for a visualization tool.

In this paper we present our *interactive NS-2 protocol and environment confirmation tool* (iNSpect)[1]. The iNSpect program was developed to allow direct visualization and analysis 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

*Research Group's URL is *http://toilers.mines.edu*. A shortened version of this paper appeared in the Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS), pp. 503–506, 2005. Email: {skurkows, tcamp, mcolagro}@mines.edu.

[1]As of December 2005, iNSpect has been requested from and shared with 105 researchers at 60 research labs/universities in 33 countries.

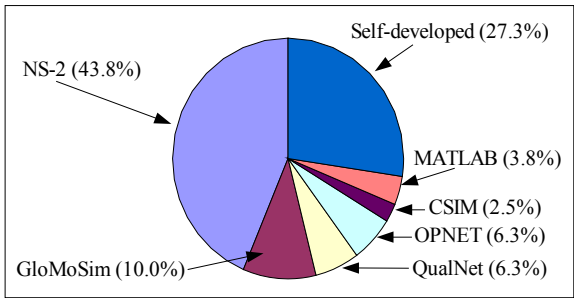


Figure 1: Simulator usage results from a survey of simulation-based papers in ACM's International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) 2000-2005 [13].

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 addition to NS-2, iNSpect can also be used with any simulator or testbed environment which produces output in the iNSpect expected file format (see Section 2.2). To see iNSpect in action, go to `http://toilers.mines.edu/iNSpect`.

1.1 Simulators and Testbeds

There are many discrete-event network simulators available for the MANET community [23]. Unfortunately, 34 of the 114 published 2000-2005 MobiHoc simulation papers (29.8%) did not identify the simulator used in the research [13]. Figure 1 shows the simulator usage results of the MobiHoc authors that did identify the simulator used. As shown, the Network Simulator-2 (NS-2) [22] is the most used simulator in MANET research; 35 of the 80 simulation papers that state the simulator used in the simulation study used NS-2 (43.8%). NS-2 was initially a wired simulator developed from the REAL network simulator [24]; the NS-2 effort has been supported by the VINT Group and NSF. The Monarch Project (from Carnegie Mellon University Rice University) has extended NS-2 to simulate wireless networks [22].

We developed iNSpect to work with NS-2 input and output files directly, because NS-2 is the most popular simulator used in the MANET community. However, as Figure 1 shows, several other simulators are used for wireless network simulations. In addition, ad hoc network testbeds (e.g., wireless sensor network testbeds) are becoming quite common. Thus, we have developed an iNSpect input option that allows iNSpect to read a specific iNSpect formatted trace file. The *vizTrace* file format allows any simulator or testbed that can generate custom output to use iNSpect.

1.2 Related Work

A visualization tool is needed to understand the large amount of data produced during network simulations. For these reasons, the Network Animator (NAM) was designed to provide a graphical user interface for the creation of wired network topologies in NS-2 [16]. 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 Project [22] to visualize wireless networking at the same level as wired networks. With the increased demand for NS-2 simulations for wireless networks, a robust visualization tool is needed for wireless networks. There was an effort in the late 1990s to develop Ad-hockey [17], a visualization tool for NS-2 wireless simulations, but that effort has not continued. The last supported update of Ad-hockey was 1999. Thus, Ad-hockey does not work with the Tool Command Language (Tcl) versions of NS-2 currently used (since version NS-2.1b7a). The current version of NS-2 is NS-2.29 [24].

More recently, since the development of iNSpect, the authors of [25] have created a 3-D visualization tool for NS-2 called *Huginn*. *Huginn* provides visualization of NS-2 trace files and filtering at different levels of the

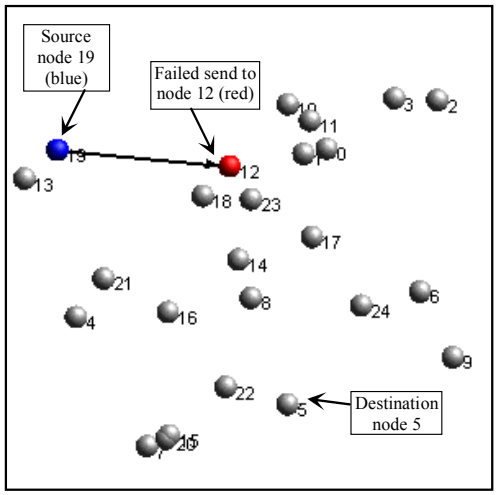


Figure 2: iNSpect showing a failed attempt to send a packet from node 19 (blue) to node 5, via intermediate node 12 (red).

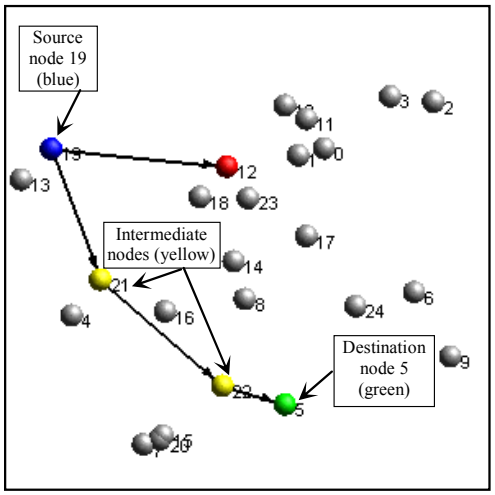


Figure 3: iNSpect showing node 19's (blue) successful attempt to send a packet to node 5 (green) via intermediate nodes 21 and 22 (yellow).

Note: Simulation area is 300 m x 300 m with 25 nodes.

network layers to generate various simulation results. However, at this time, *Huginn* has not been released to the community.

2 iNSpect Overview

The *interactive NS-2 protocol and environment confirmation tool* (iNSpect) is a C++ OpenGL-based [28] visualization tool that allows analysis of simulated wireless networks. The iNSpect program uses a GTK+ [8] graphical user interface (GUI) for direct scene manipulation. The iNSpect program is multi-platform and will execute on Linux, MacOS X, Windows, and Cygwin.

2.1 iNSpect Visualization

The iNSpect program produces a 3-dimensional visual display of the nodes in a wireless scenario based on Cartesian (x,y,z) coordinates used by NS-2. Unlike NAM, which does not show the transmissions in wireless networks, the iNSpect program shows the wireless routes and the success or failure of wireless packet transmissions. 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 events associated with a node are mapped to colors by a configuration file. A customizable color scheme aids in analysis.

Figure 2 illustrates the basic use of iNSpect, with the default color scheme (i.e., blue sending nodes, yellow forwarding nodes, green destination nodes, and red unsuccessful transmissions). In Figure 2, node 19 is attempting to send a packet to node 5 via intermediate node 12, but node 12 does not receive the packet. In Figure 3, node 19 successfully sends the packet to node 21, which forwards the packet to node 22, which forwards the packet to the destination node 5. The persistence of the lines and status of the packet activity is configurable, allowing for individual route analysis.

In summary, with the default color scheme, blue and yellow nodes along a path leading to a green node shows a successful transmission to a destination node; blue and yellow nodes along a path leading to a red node shows

Table 1: Format and example events for iNSpect's *vizTrace* file.

Field and Description					
Node	Time	Event	Other Node	Status	Packet
ID	Seconds elapsed	*sending to* or *received from*	*ID* or *-1*	Custom string	ID
Sample data					
2	2.34628	sending to	6	source	45
6	2.35677	received from	2	forwarding	45
6	2.35999	sending to	12	forwarding	45
12	2.36125	received from	6	destination	45

failure of the packet to reach the destination node and at which hop the packet failed. A graphical representation of network activity gives the researcher more clues about the individual success and failure of packets than the overall delivery ratio printed at the end of a scenario.

2.2 iNSpect Input

The iNSpect program uses an event builder object to schedule node movement and packet traffic of various types and formats. Because the event builder can translate different inputs, the iNSpect program can animate a stand-alone mobility file (an NS-2 input file), an NS-2 trace file (an NS-2 output file), and a mobility file with a specific iNSpect formatted trace file (a *vizTrace* file).

2.2.1 Stand-alone Mobility File

Our iNSpect program can be used directly with a mobility file generated by an external mobility model. Unlike NAM, there is no requirement to first generate a trace file from NS-2. Mobility file analysis outside of NS-2 is extremely useful for mobility model development and mobility model output verification, eliminating the overhead of additional lengthy simulations in NS-2 to generate a trace file.

2.2.2 NS-2 Trace File

For protocol evaluation, the NS-2 generated wireless trace file can be used by the iNSpect event builder to schedule packet transmissions and to process node movements without a mobility file. An NS-2 trace file created in the Tool Command Language (Tcl) driver file with medium access control (MAC) layer trace (`macTrace`) and movement trace (`movementTrace`) turned on contains all of the node packet and movement activity. The iNSpect program can process this stand-alone NS-2 trace file without using an externally generated mobility file.

2.2.3 Mobility File and vizTrace File

The iNSpect trace file, called a *vizTrace* file, allows iNSpect to be used with any simulator or testbed that produces a trace file in the iNSpect expected format. To build a *vizTrace* file, a researcher records each event in the format of Table 1. (The options for an entry are listed below the name of each field.) The six fields of each event are included in each line of the *vizTrace* file: node ID, event time, event title, other node ID, status of event, and packet ID. The user defined *status* of event field enables customized color schemes and statistics (see Section 3.3.2).

Table 1 also shows four example events in a *vizTrace* file. The example events show node 2 sending packet number 45 to node 12 via node 6. In the user defined *status* field, node 2 is the *source*, node 6 is the *forwarding*

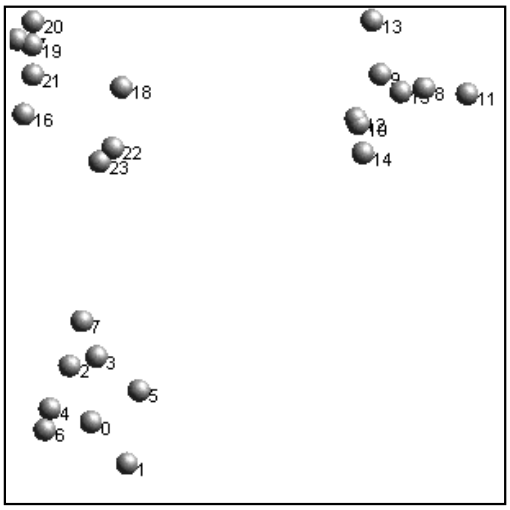


Figure 4: iNSpect displaying the RPGM model with 60 m reference point separation.

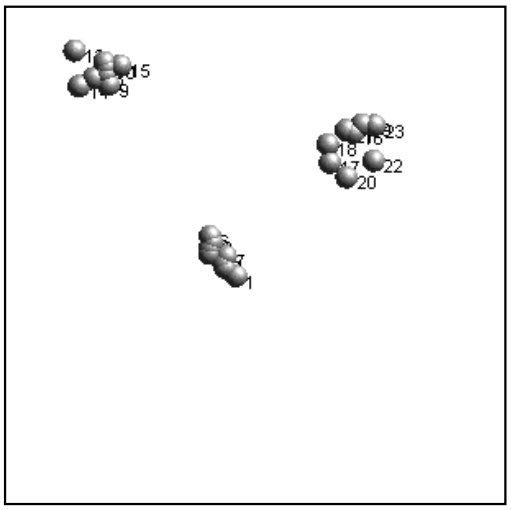


Figure 5: iNSpect displaying the RPGM model with 15 m reference point separation.

Note: Simulation area is 300 m x 300 m with 3 groups of 8 nodes each, 20 m/s node speed, and 0 seconds of pause time.

node, and node 12 is the *destination* for packet number 45. These user defined terms can then be associated with specific colors in iNSpect. In summary, iNSpect can process any simulator or testbed output file if the output file is in the *vizTrace* format.

3 iNSpect Uses and Results

In this section we highlight our successes with iNSpect as a research tool. We have used iNSpect to develop, analyze and verify mobility models, to find a problem in NS-2.27, to verify protocol development, and to analyze protocol performance.

3.1 Topology Analysis and Validation

Visualizing nodes moving can help verify a mobility model. With NS-2, the complete analysis of a mobility file can be done only by running a simulation through NS-2 to produce a NAM trace file. NAM would then be used to visualize the NAM trace file, and ultimately the node movement.

Unlike NAM, iNSpect can process an NS-2 based mobility file directly. The iNSpect engine calculates the node movements directly from the mobility file. This capability streamlines the development of mobility files generated by 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. Thus, iNSpect can produce visual verification for a mobility file instantly. The direct processing of the mobility file allows a developer to complete many iterations quickly.

Figures 4 and 5 illustrate an example verification of the Reference Point Group Mobility (RPGM) Model [4, 21]. To generate a mobility file from the RPGM model, the user must determine numerous parameters including reference point separation distances and individual node separations from the reference point [4]. Figures 4 and 5 illustrate how a user can analyze these two parameters in iNSpect. Figures 4 and 5 show two mobility files with the same dimensions, speed, pause time, number of nodes, and number of groups, but different levels of node separation from the reference point. Immediately the effect of the parameter is seen. The iNSpect program is the

only way to see the effect of these parameters from a mobility file directly.

Furthermore, because iNSpect allows the immediate verification of files produced by a mobility model, iNSpect can be used to develop new mobility models. For example, we used iNSpect during the development of a new congestion-based mobility model. In this new 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 movement problem with the implementation of the model. With iNSpect, our implementation problem was debugged and quickly fixed. Without iNSpect the problem might have gone unnoticed.

3.2 Simulation Model Analysis and Validation

As stated at the beginning of the NS-2 documentation [24] *"users of NS-2 are responsible for verifying for themselves that their simulations are not invalidated by bugs."* The question is how one ensures a simulation is correct? While there is no way to guarantee correctness, iNSpect can help. The iNSpect program can provide insight into the simulation process that summary statistics cannot provide.

As an example, when we upgraded to NS-2 version 2.27, we noticed a significant drop in the performance of our simulations (e.g., delivery ratio), similar to several accounts on the NS-mailing list (e.g., [5]). 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 [26].) The NS-2.27 error is shown in Figure 6. In Figure 6, the simulation area is 600 m x 600 m. Each node's transmission range is 100 m. As shown, node 0 successfully transmits a packet outside the 100 m range (e.g., node 0 to node 26 in Figure 6). The actual distance between node 0 and node 26 is 453 meters, which is well over the 100 m transmission range. The blue star in Figure 6 shows NS-2.27's incorrect view of node 0's location, explaining the reason NS-2 allowed the transmission to be successful.

In summary, iNSpect quickly illustrated the inconsistencies of the simulation output under 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 (e.g., the incorrect location of node 0 in Figure 6). Second, NAM does not show the links and packet flows. Thus, even if the nodes were in their correct locations, one would not have seen the transmission over 100 m that succeeded.

3.3 Simulation Results Analysis

An entire simulation (node movement and wireless network traffic) can be animated with iNSpect. The iNSpect display shows each transmission, with lines between nodes for transmission attempts and node colors which represent 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. The ring, although representative of an omni-directional wireless signal without obstacles, does not help a researcher trace the route of a packet. The iNSpect animation allows quick analysis of packet routes and node activities. An animation of the results aids understanding of summary performance statistics such as delivery ratio, end-to-end delay, and overhead.

3.3.1 Path Analysis

By knowing the path a packet takes from source to destination, we can learn more about the behavior of a protocol. Figure 7 shows a snapshot of a Location Aided Routing (LAR) simulation [12]. 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 7 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 13 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 the protocol did not discover a shorter route. Individual analysis such as this would be impossible with

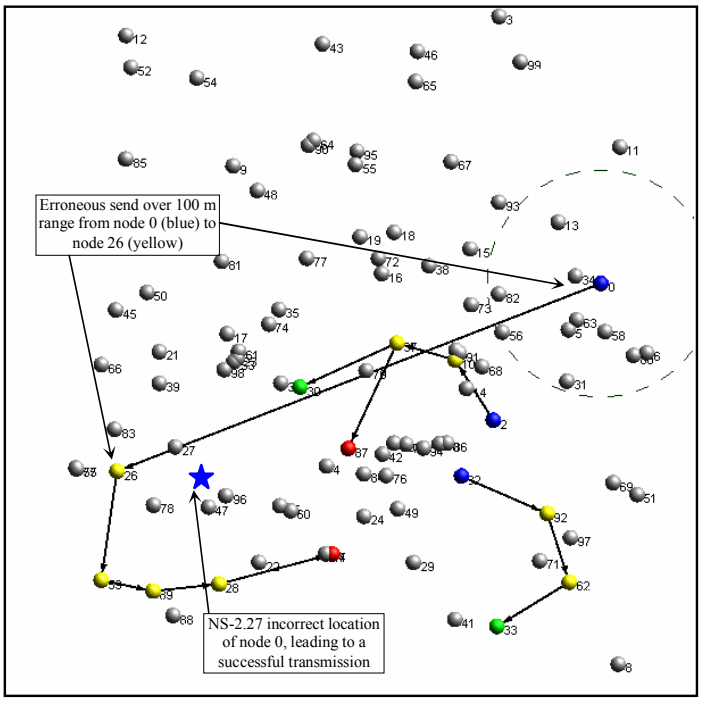


Figure 6: iNSpect showing an NS-2.27 model error. Node 0's transmission exceeds the 100 m transmission range, because NS-2's incorrect view of node 0's location places node 0 in range of node 26 (blue star). The simulation area is 600 m x 600 m with 100 nodes.

only performance statistics and no iNSpect visualization. With the help of iNSpect, we developed improvements to LAR that utilize the location information disseminated to find more direct routes [6].

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 8 shows an example route discovered in our LAR projection method. Our LAR improvement found a route with eight fewer hops, compared to the route shown in Figure 7, for the same transmission (from node 5 to 76, 19, 92, 77, to 44). The iNSpect program visualizes the reduced number of hops, and that the resulting path is the shortest between node 5 and node 44. Visualizing the routes with our LAR projection method is a valuable step in designing and analyzing our improvements to the LAR protocol.

3.3.2 Node Activity Analysis

By knowing the nodes' activities, we can use iNSpect's custom color capability to visualize and monitor the network activity at the nodes. For example, in a simple flooding protocol, packets are broadcast to all of a node's neighbors, all of which rebroadcast the packet [27]. There are, of course, several improvements to simple flooding; see [27] for a comparison of broadcasting protocols. One variation of an improved flooding protocol is the Probabilistic Broadcasting protocol [20]. In the Probabilistic Broadcasting protocol, a node rebroadcasts with probability (P), in order to reduce duplication and collisions. Figures 9 and 10 show a custom color scheme comparing simple flooding and the Probabilistic Broadcasting protocol for the same scenario. The node colors in these two figures were defined in the user defined *status* fields, as shown in Table 2.

In Figures 9 and 10, we see the transmit lines and color coded nodes for a simple flooding and Probabilistic

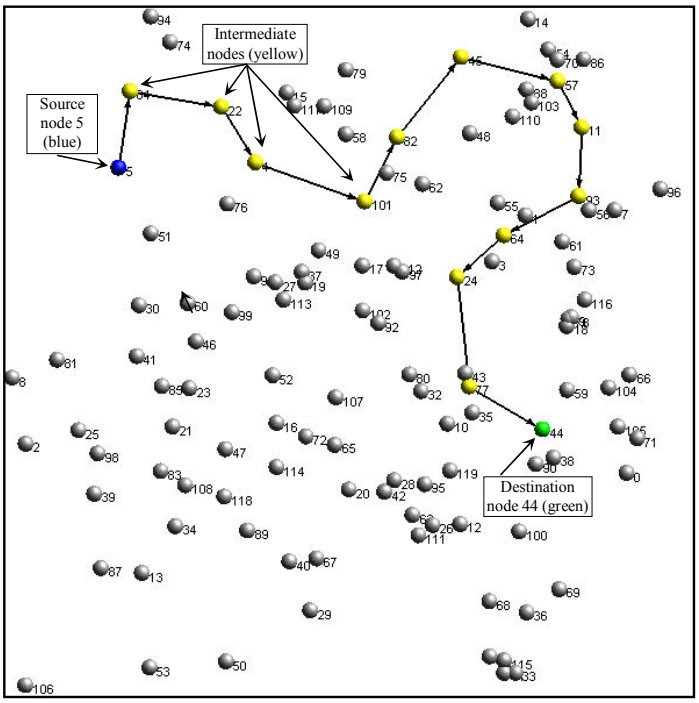


Figure 7: iNSpect showing a Location Aided Routing route selection for a transmission from node 5 to node 44 in a 600 m x 600 m simulation area with a 100 m transmission range and 120 nodes.

Broadcast ($P = 0.38$) transmission from node 2 at the instant the final node (4, green) receives the packet. The iNSpect display immediately shows the difference between the simple flood and the Probabilistic Broadcast protocols (e.g., no red nodes exist in Figure 10). As shown, the Probabilistic Broadcast protocol, compared to simple flooding, sends the packet to all nodes with fewer transmissions (i.e., less lines exist between the nodes) and fewer duplicate packets (i.e., more cyan nodes and no red nodes exist). Visualizing the node activity for simple flooding and the Probabilistic Broadcast protocol is a valuable step in understanding the performance of these two protocols.

Because iNSpect's status field is so flexible, other visualizations of the same data are possible. For example, iNSpect can color a node based on whether it rebroadcast the packet rather than how many packets it received. Visualizing which node rebroadcasts a packet is useful when designing a protocol that approximates the minimum connected dominating set. When designing a protocol that uses neighbor knowledge, iNSpect can color a node

Table 2: Status field values for Figures 9 and 10.

Value (color)	**Description**
source (blue)	Initiated the packet transmission
final (green)	Last node to receive the packet
received (cyan)	Received the packet once
duplicate (orange)	Received the same packet twice
2-duplicates (pink)	Received the same packet at least three times
4-duplicates (red)	Received the same packet at least five times

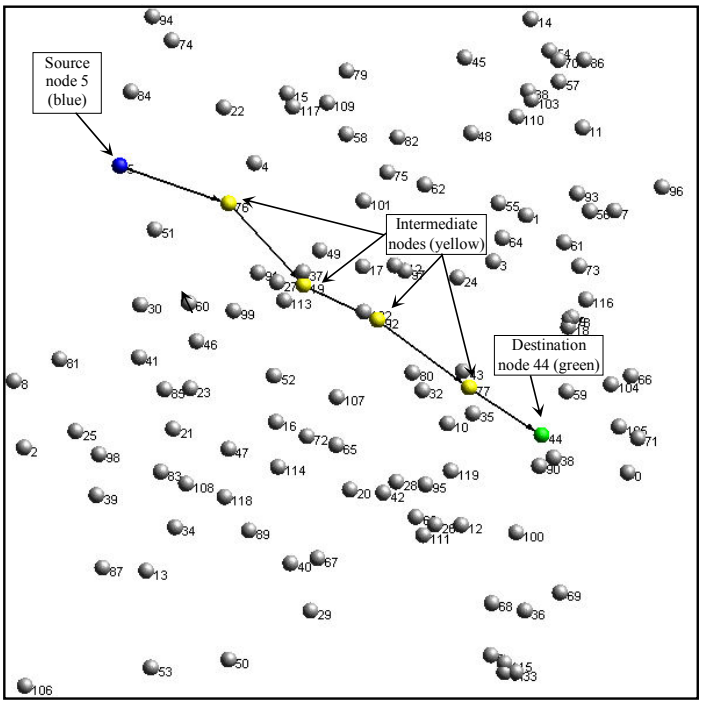


Figure 8: iNSpect showing our projection method [6] for the Location Aided Routing protocol. The same scenario as Figure 7 is used.

based on the number of one-hop neighbors or two-hop neighbors. Because iNSpect allows the researcher to define the visualization, the most useful data can be highlighted.

3.3.3 Performance Analysis

With the suite of calculations that iNSpect provides, researchers now have more analytical techniques available. For example, we executed a protocol with the same scenario multiple times and each time it produced significantly different delivery ratios. Using iNSpect's connectivity graph and partition check tools, which are described in Section 4.1, we found the scenario had a large group of nodes isolated from the rest. As a result, when the randomly selected source and destination nodes were in the same non-partitioned set, the delivery ratio was higher than when the source and destination nodes were split across the partition. The iNSpect program made it possible to quickly understand the differing performance statistics.

4 iNSpect Details

The iNSpect program offers several calculations and design features that provide the researcher with a powerful visualization and analysis tool. In this section we discuss details of the iNSpect program.

4.1 iNSpect Calculations

4.1.1 Connectivity Graph

The *Connectivity Graph* tool renders a line between nodes that are within range of each other based on the transmission range of each node. Figure 11 shows the iNSpect program with the connectivity graph illustrated.

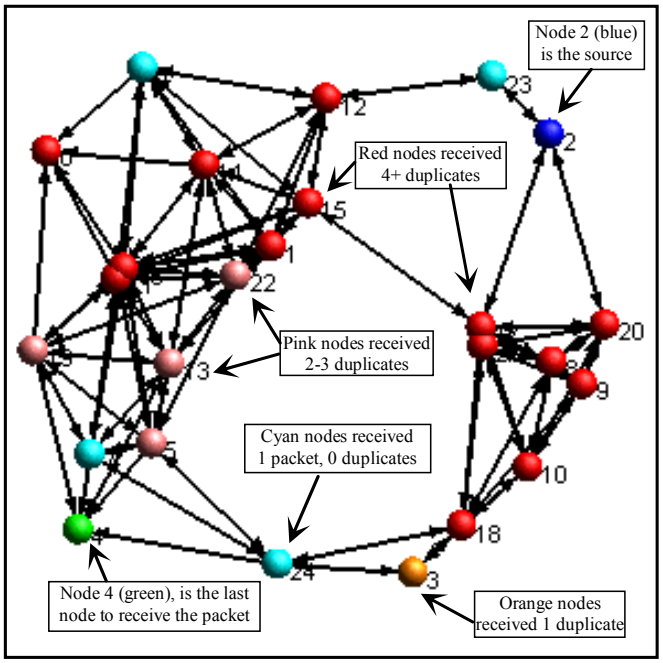


Figure 9: iNSpect displaying a snapshot of a simple flooding protocol, at the time when the last node (node 4, green) successfully received the packet. Node 2 (blue) is the source of the broadcast packet.

The connectivity lines can be used to determine shortest paths, unavailable paths, and potential routing loops. The paths can also be compared to the node's neighbor tables, to determine the accuracy and currency of each node's neighbor information.

4.1.2 Partition Check

The *Partition Check* tool identifies isolated nodes in a network. Partitioning occurs when a node is not connected with any other node in the network and, when present, can impact protocol performance. The *Partition Check* tool changes the appearance of any node that is disconnected from the rest of the nodes in the network. Figure 11 shows node 1 is partitioned from the other nodes in the network. We use the *Partition Check* tool to check the degree of partitioning present in a network. Generating adjacency and connectivity matrices to check a scenario for partitioning is an expensive calculation. However, because iNSpect is already scanning and rendering each node, visualizing partitioning is an inexpensive calculation during the playback.

4.1.3 Node Reports

The iNSpect program can generate reports of the performance statistics calculated during the simulation. These reports can be generated anytime throughout the playback. A *Node Report* is shown in Figure 12 and includes the total number of packets delivered (destination) to the node, dropped by the node, forwarded by the node, and sent (source) by the node, for each node in the simulation. The *Node Report* provides more than a summation of the trace file entries. For example, the NS-2 trace file contains only send, receive, and collision packet events. Packets that are sent from a node and never received at the next hop are not recorded as dropped packets. The iNSpect program links a targeted node with a send attempt. This link allows iNSpect to calculate the number of packets forwarded, dropped, and received successfully at the destination and to provide this information in the

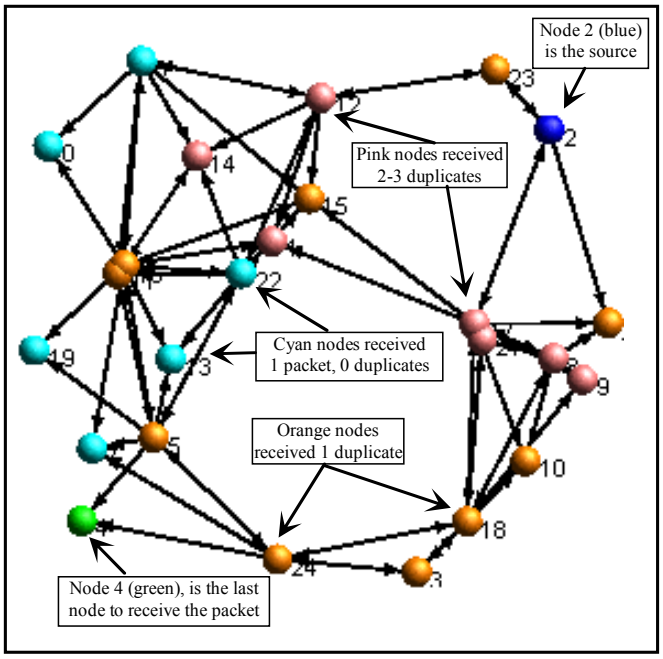


Figure 10: iNSpect displaying a snapshot of the Probabilistic Broadcast protocol, for the same scenario as Figure 9, at the time when the last node (node 4, green) successfully received the packet. Node 2 (blue) is the source of the broadcast packet.

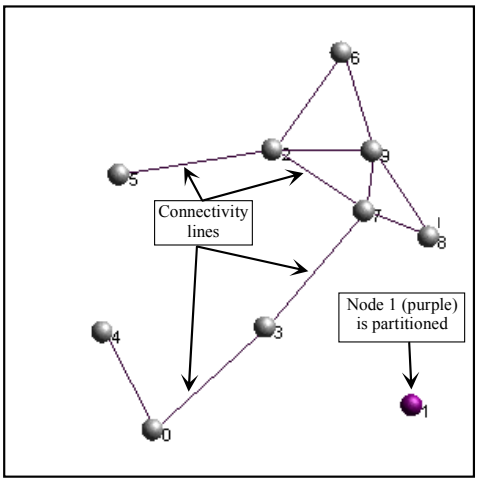
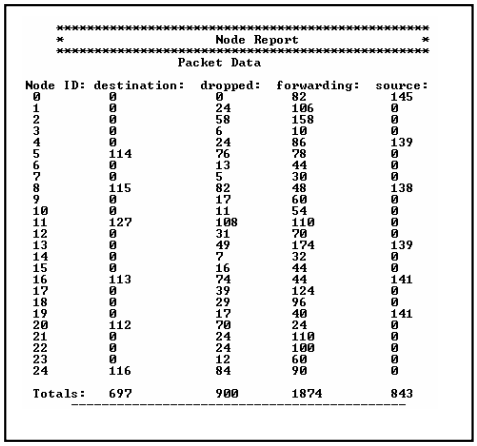


Figure 11: iNSpect showing the connectivity graph and partition check calculations for a 300 m x 300 m simulation area with 10 nodes. Transmission range is 100 m.

Node Report. The *Node Report* can then be used to identify any unusual trends in certain nodes or areas of the scenario.



****
* Node Report *
****

Packet Data

Node ID:	destination:	dropped:	forwarding:	source:
0	0	0	82	145
1	0	24	106	0
2	0	58	158	0
3	0	6	10	0
4	0	24	86	139
5	114	76	78	0
6	0	13	44	0
7	0	5	30	0
8	115	82	48	138
9	0	17	60	0
10	0	11	54	0
11	127	108	110	0
12	0	31	70	0
13	0	49	174	139
14	0	7	32	0
15	0	16	44	0
16	113	74	44	141
17	0	39	124	0
18	0	29	96	0
19	0	17	40	141
20	112	70	24	0
21	0	24	110	0
22	0	24	100	0
23	0	12	60	0
24	116	84	90	0
Totals:	697	900	1874	843

Figure 12: An iNSpect *Node Report* showing the number of packets received, dropped, forwarded, and sent by each node.

4.2 iNSpect Features

The iNSpect program provides a feature rich environment to visualize and analyze wireless networks. The iNSpect program has display maneuvering, node locations, node selection, and the following overlay capabilities: geometric shapes and background images. The iNSpect program also provides direct image and movie capture. Additionally, all customizable settings are configurable through the iNSpect control panel or configuration files. We discuss iNSpect's features in this section.

4.2.1 Display Manipulation

The iNSpect display area is a three dimensional rendering area for OpenGL, that allows the researcher to zoom in and out. The display area can also be panned up, down, left, and right. In addition, there is a slider bar and buttons to adjust the time of the simulation playback, allowing the user to move the simulation time forward or backward (see Figure 13).

4.2.2 Node Location

The iNSpect coordinate overlay displays node locations in (x,y) coordinates; see Figure 14 for an example. Location information can 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 [15]. 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 summary statistics.

4.2.3 Node Selection

Clicking on a node in the display selects the node. When an individual node is selected in the scenario, a transmission range ring (dotted line) is displayed showing the ideal transmission range of the node (see Figure 13, node 45, and Figure 14, node 21). Additionally, the *Node Status* window (see Figure 13) is updated with the node's number, the node's current x and y location, and the node's current status from the trace file (e.g., source or destination). The *Node Status* window also gives current totals for the different status types in the trace file.

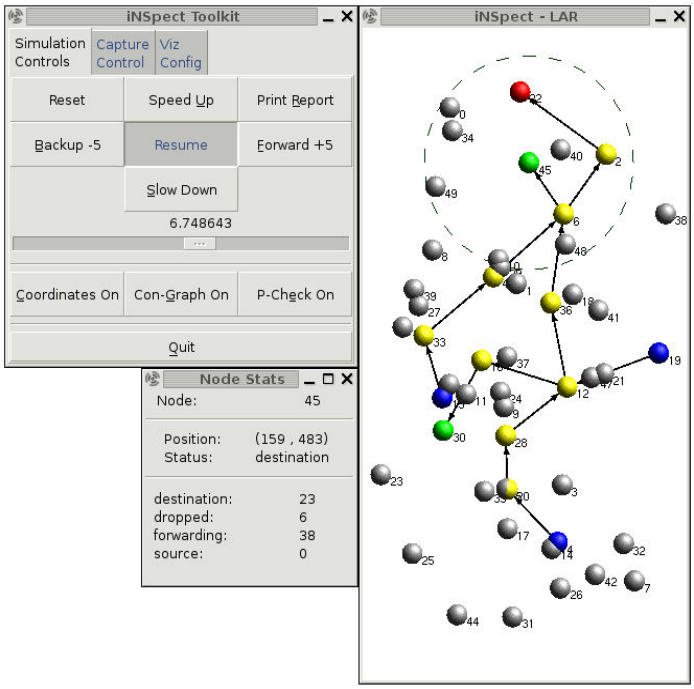


Figure 13: The iNSpect graphical user interface. Clockwise from top left: iNSpect control window, network display window and the node status window.

4.2.4 Geometric Shapes

The iNSpect program allows a user to display geometric objects, such as circles and rectangles, which may identify regions of interest. We used circular overlays 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 [4]. (The nodes are initialized in the steady state distribution of the Random Waypoint Mobility Model [18, 19].) Then, as a node moves into the area of congestion, the node slows down. We use the circular overlay in iNSpect to represent the congested area, verifying that the nodes slow in this defined area. The area can represent a food court at a mall or a large intersection in a city. The location and size of the circular objects are configurable within iNSpect.

A rectangular overlay is also available in iNSpect. We used a rectangular overlay 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 14 depicts a rectangular area of interest with corners at (200 m, 500 m) and (300 m, 600 m). The representation of the rectangle on the 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 [1], 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.

4.2.5 Background Image Display

The iNSpect program allows the researcher to display background images. The GTK+ toolkit enables iNSpect to support popular image formats (`jpeg`, `gif`, `png`, etc.). The background in the rendering area is an image. While the default image is white, other images can be loaded for display. Figure 15 shows an example of a scenario

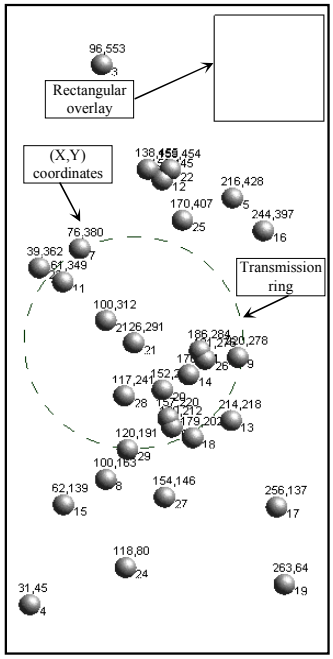


Figure 14: iNSpect showing node location, transmission ring, and square area of interest in a 300 m x 600 m simulation area with 30 nodes. Transmission range is 100 m.

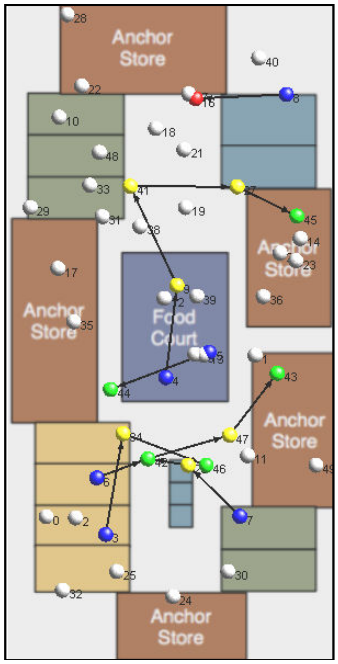


Figure 15: iNSpect visualization with a mall as an overlayed background image. Simulation area is 0.5 km x 1 km with 50 nodes.

with a shopping mall map loaded as the background. The image display capability allows a researcher to display networks and nodes on a real background, adding context to scenarios for education and presentation purposes.

4.2.6 Image/Movie Capture

The iNSpect program has native screen capture and movie capture capability for presentations and education. The *Capture Control* tab provides facilities to capture screenshots of the display area in both `ppm` and `png` formats. The images are captured directly from the frame buffer for high quality images. The *Capture Control* tab also provides controls to produce `MJPEG` encoded movies. The movie control provides a selector to set the start time, stop time, and frame rate. These screenshots and images can be used for education and presentation purposes. We used these controls to capture the images for this paper.

4.3 iNSpect Implementation

4.3.1 Graphical User Interface

The iNSpect program provides a graphical user interface (GUI) for researchers to interact effectively with the playback environment. The *Simulation Controls* tab of the iNSpect GUI is illustrated in Figure 13. The "Speed-up" button doubles the simulation playback speed and the "Slow-down" button halves the simulation playback speed. The "Backup -5" and "Forward +5" buttons move the simulation playback timer backward or forward five seconds, respectively. The Pause/Resume button works as one would expect; Figure 13 shows an example of the *Simulation Controls* tab in the paused state. The slider bar allows the user to move to any point (forwards or backwards) in the simulation. The current simulation time is displayed above the slider bar. The three buttons above Quit (i.e., "Coordinates On", "Con-Graph On", and "P-Check On") were discussed in Section 4.1. Finally,

all controls can be accessed using a hot key from both the toolkit and simulation windows. As an example, the 'p' key can be pressed once to pause the simulation and then again to resume the simulation.

4.3.2 Configuration File

Our iNSpect program is driven by a configuration file, which minimizes the command line arguments while enabling the user to control numerous aspects of the display. All user configurable parameters are defined by defaults in the program or by values provided in the configuration file. The configuration file and system defaults minimize the amount of effort required by a researcher to customize iNSpect for his or her needs. For example, the user can define the start and end time of the playback, which allows a researcher to jump to a specific portion of the playback quickly. If the user does not define the start and end time, the playback will begin at zero seconds and end after the full trace file is played. These values can be changed directly in the configuration file, or the *Viz Config* tab can be used to adjust and save configurable items. Using the *Viz Config* tab allows a researcher to see the immediate impact of a parameter change.

4.3.3 File Parser

Input/output processing can be a performance issue for NS-2 simulation playback due to the large size of NS-2 trace files. (A typical NS-2 simulation can generate trace files over 1 GB for a 1000 second simulation.) The iNSpect program utilizes a threaded parser and a read-ahead scheme to keep data flowing to the display portion of iNSpect. The iNSpect program starts the file parser early in the startup sequence. The parser signals the display to begin rendering when the parser has read sufficient data for each node in the display. The file parser then continues to read-ahead in the background while the display is rendering the scenario to the researcher. The file parser is implemented as a thread to eliminate blocking between the file read and display. This approach enables the researcher to view the NS-2 simulation scenario quickly and smoothly, and avoids a several minute wait to pre-process a large trace file.

4.4 Additional uses

The iNSpect program can also be used to verify propagation models and transmission range behavior. In this case, the researcher places nodes at varying distances around a test node and has the test node 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.

Furthermore, because iNSpect is C++, GTK+, and OpenGL code, it is easy to write front-end processing units. 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.

5 Conclusions

With the increase in wireless network research, visualization and analysis of node behavior, simulations, and results are necessary to engage in productive development. By using the iNSpect tool, a researcher can discover anomalies in topology files, the simulation model itself, or even in the results of a particular protocol. As we have seen in our own research, iNSpect can reveal issues that summary statistics cannot and has saved us hours of detailed detective work trying to verify results. 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 works directly with NS-2 input and output files, and can read a specific iNSpect formatted trace file from other simulators or testbeds. In short, iNSpect lets the human visual system to participate in the analysis of wireless simulation results. For details on obtaining iNSpect, go to `http://toilers.mines.edu/iNSpect`.

6 Acknowledgments

This work was supported in part by NSF Grants ANI-0208352 and ANI-0240558. 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. We also thank Neil "Tuli" Mushell, Matthew Gimlin, and Neal Erickson for their efforts in coding, debugging, and improving iNSpect.

References

[1] K.C. Almeroth, A. Jardosh, E. M. Belding-Royer, and S. Suri. Towards realistic mobility models for mobile ad hoc networks. In *Proceedings of the Nineth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 217–229, 2003.

[2] K. Amallada. NAM with wireless simulation. http://mailman.isi.edu/pipermail/ns-users/2004-April/041770.html. Page accessed on December 21, 2005.

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

[4] 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.

[5] J. Chen. DSR performance is too bad in NS-2, why? http://mailman.isi.edu/pipermail/ns-users/2004-March/040565.html. Page accessed on December 21, 2005.

[6] 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)*, pages 924–929, 2004.

[7] Association for Computing Machinery. The ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc). URL: http://www.sigmobile.org/mobihoc. Page accessed on December 21, 2005.

[8] The GNU image manipulation program GIMP toolkit. http://www.gtk.org. Page accessed on December 21, 2005.

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

[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. 6 pages.

[11] H. Kee. NAM support for wireless traffic. http://mailman.isi.edu/pipermail/ns-users/2004-May/042046.html. Page accessed on December 21, 2005.

[12] 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)*, pages 66–75, 1998.

[13] S. Kurkowski, T. Camp, and M. Colagrosso. MANET simulation scenarios: The incredibles. *ACM Mobile Computing and Communications Review (MC2R)*, 9(4):50–61, October 2005.

[14] J. Malek. Trace graph - network simulator ns-2 trace files analyser. URL: http://www.trace graph.com. Page accessed on December 19, 2005.

[15] M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *IEEE Network*, 15(6):30–39, 2001.

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

[17] Monarch Project: Ad-hockey for Perl/Tk800.015. http://www.monarch.cs.rice.edu/cmu-ns.html. Page accessed on December 19, 2005.

[18] W. Navidi and T. Camp. Stationary distributions for the random waypoint mobility model. *IEEE Transactions on Mobile Computing*, 3(1):99–108, 2004.

[19] 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)*, pages 319–326, 2004.

[20] S. Ni, Y. Tseng, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 151–162, 1999.

[21] 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, 1999.

[22] The CMU Monarch Project. The CMU monarch extensions to the ns simulator. URL: http://www.monarch.cs.cmu.edu/. Page accessed on December 21, 2005.

[23] The SEACORN Project. SEACORN simulation tools. URL: http://seacorn.ptinovacao.pt/ sim_tools.html. Page accessed on December 21, 2005.

[24] The VINT Project. The network simulator - ns-2. URL: http://www.isi.edu/nsnam/ns/. Page accessed on December 21, 2005.

[25] B. Scheuermann, H. Füßler, M. Transier, M. Busse, M. Mauve, and W. Effelsberg. Huginn: A 3D Visualizer for Wireless ns-2 Traces. In *Proceedings of the 8th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 134–150, 2005.

[26] K. To. Bug in ns-2.27 wireless channel. http://mailman.isi.edu/pipermail/ns-users/2004-April/041388.html Page accessed on December 21, 2005.

[27] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 194–205, 2002.

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