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

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

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 (MANETs), and sensor networks; however, the Network Animator (NAM) for NS-2 has not been extended for wireless visualization. In this paper, we discuss a new visualization and analysis 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 results of NS-2 simulations. Our iNSpect program handles 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.

1. Introduction and Related Work

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 [7]. Often, wireless research will involve a testbed implementation and/or a simulation study. The most popular simulator used by the mobile ad hoc network researchers is the Network Simulator-2 (NS-2) [19] with the Monarch project wireless extension [16] (see Figure 1). Specifically, 28 of the 63 simulation-based papers published in the 2000-2004 ACM MobiHoc proceedings (44.4%) use NS-2 [12]. NS-2 was initially a wired simulator developed from the REAL network simulator [19]; the NS-2 effort is supported by the VINT Group and NSF.

*This work supported in part by NSF Grants ANI-0208352 and ANI-0240558. Research Group's URL is *http://toilers.mines.edu.*

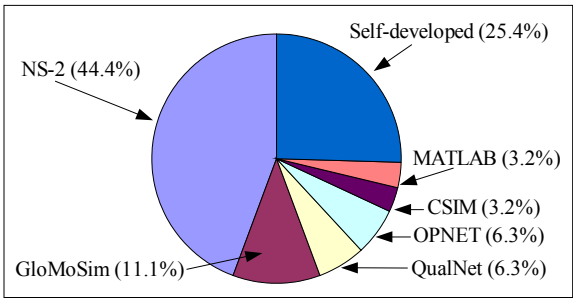


Figure 1. Simulator usage results from a survey of simulation-based papers in ACM's International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) 2000-2004 [12].

A visualization tool is needed to understand the large amount of data produced during network simulations. 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 [1]. Visualizations add to the understanding gained via statistical analysis. For these reasons, the Network Animator (NAM) was designed to provide a graphical user interface for the creation of wired network topologies [14]. 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. Currently, NAM displays only the node positions and movement in the network. The only communications visualized in NAM are energy pulses transmitted from a node when it sends a packet. NAM does not provide packet flow nor monitoring.

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 [15], 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-2.28.

There have been many emails on the NS-users mailing list asking for visualization or video support for wireless networks in NS-2 (see [6] and [10], for examples). The increasing complexity of node and protocol behavior is driving the need for a visualization tool.

In this paper we introduce our *interactive NS-2 protocol and environment confirmation tool* (iNSpect). The iNSpect program allows the 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 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. To see iNSpect in action, go to `http://toilers.mines.edu/iNSpect` [1].

2. iNSpect Overview

The *interactive NS-2 protocol and environment confirmation tool* (iNSpect) is a C++ OpenGL-based [22] visualization tool that allows analysis of wireless networks simulated in NS-2. The iNSpect program uses a GTK+ [5] graphical user interface (GUI), for direct scene manipulation. The iNSpect program is multi-platform and will execute on Linux, Windows, MacOS X, and Cygwin.

The iNSpect program can animate both a single standalone mobility file (an NS-2 input file) and an NS-2 trace file (an NS-2 output file). The iNSpect program produces a visual display of the nodes in a wireless scenario based on Cartesian (x,y) 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 a mobility file generated by external mobility models. Unlike NAM, 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 simulations in NS-2 to generate a trace file.

The NS-2 simulator generates a trace file that can be used by the event builder to schedule packet transmissions and process node movements. 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 initiating node of a transmission attempt turns blue to indicate it is sending in iNSpect. 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 indicate the next forwarding transmission attempt, and the route line is extended to show the forwarding of the packet. Figures 2–4 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 is configurable and allows for individual route analysis.

In Figure 2, node 1 initiates its packet send to node 48, by attempting to send the packet to node 20. The iNSpect program represents the transmission 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 3). 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 4). Figure 4 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.

3. iNSpect Uses and Results

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

3.1. Topology Analysis and Validation

A visualization of the nodes moving can help validate a mobility model. Until the development of iNSpect, the complete analysis of a mobility file could only be done by running a simulation through NS-2 to produce a NAM trace file. NAM would then be used to visualize the NAM trace file.

[1] As of April 2005, iNSpect has been requested from and shared with 47 researchers at 42 research labs/universities in 16 countries.

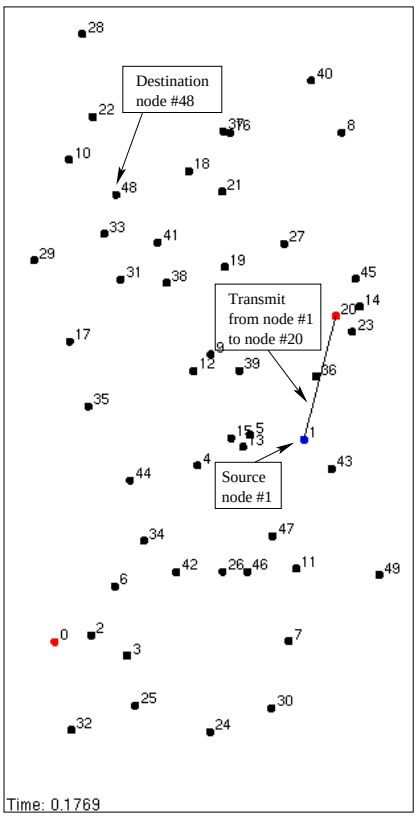


Figure 2. iNSpect showing the first hop of node 1 transmitting to node 48.

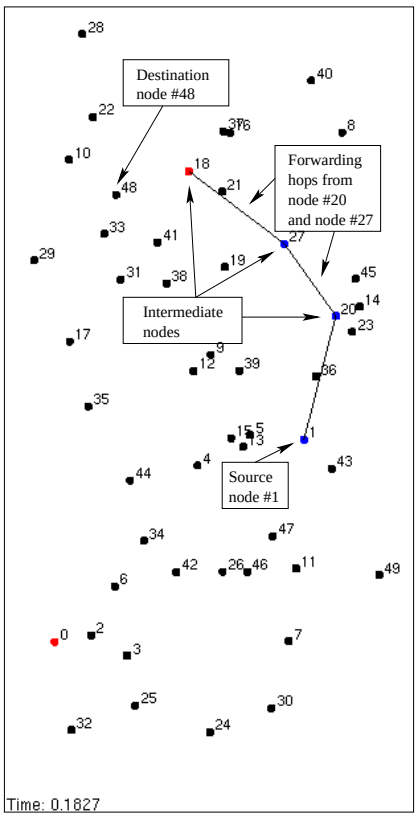


Figure 3. iNSpect showing the forwarding of the packet from node 20 through nodes 27 and 21 to node 18.

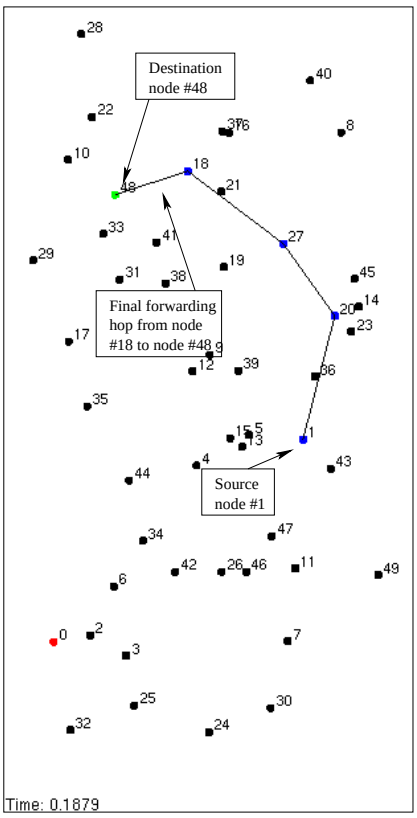


Figure 4. iNSpect showing the final hop of the route from node 1 to destination node 48.

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

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 validation for a mobility file instantly. The direct processing of the mobility file allows a developer to complete many iterations quickly.

Figures 5 and 6 illustrate an example validation of the Reference Point Group Mobility (RPGM) Model [2], [20]. To generate a mobility file from the RPGM model, the user must determine numerous parameters including reference point separation distances and individual node wanderings from the reference point [2]. Figures 5 and 6 illustrate how a user can analyze these last two parameters in iNSpect. Figures 5 and 6 show two mobility files with the same dimensions, speed, pause time, number of nodes, and number of groups, but different 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 effect of these parameters from a mobility file directly.

Furthermore, because iNSpect allows the immediate validation 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 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 [19] "*users of NS-2 are responsible for verifying for them-*

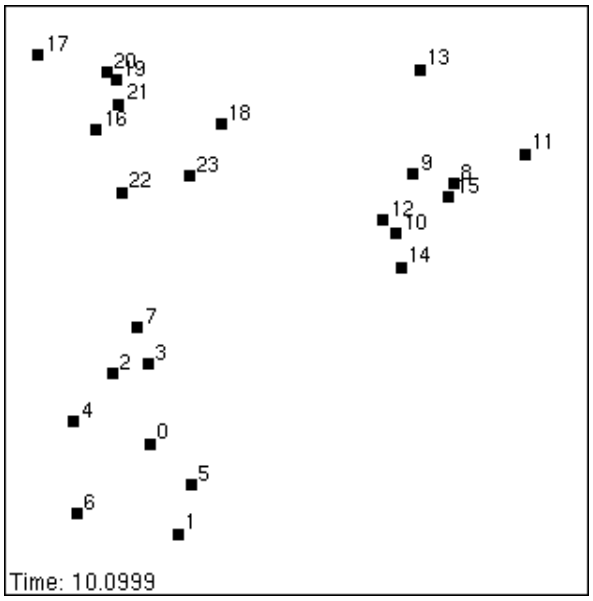


Figure 5. iNSpect displaying an RPGM model example with loosened travel from reference point.

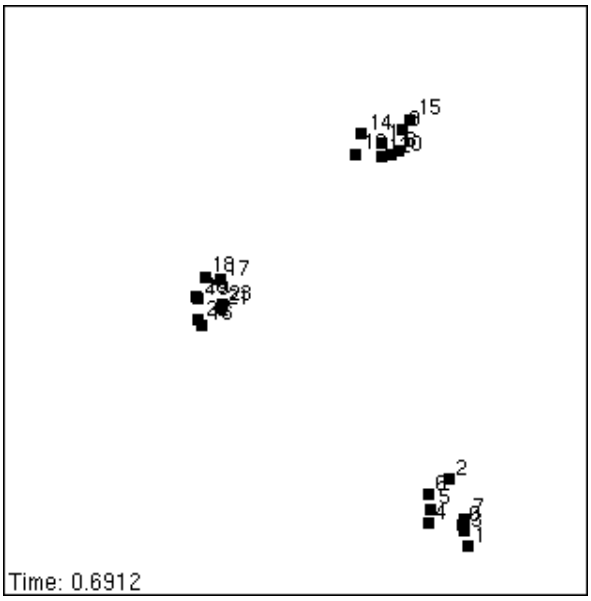


Figure 6. iNSpect displaying an RPGM model example with restricted travel from reference point.

Note: Simulation area is 300 m x 300 m with 3 groups of 8 nodes each

selves 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. iNSpect 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., [3]). Using iNSpect, we discovered an error in the simulator. Specifically, NS-2.27 did not update the position of a node unless there is an event for that node. (The error was concurrently located by the author of [21].) The NS-2.27 error is shown in Figures 7 and 8. In Figure 7, the simulation area is 300 m x 600 m and in Figure 8, 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 7), (node 16 to node 8 in Figure 7), and (node 0 to node 26 in Figure 8). In Figure 8, the actual distance between node 0 and node 26 is 453 meters, which is well over the 100 m transmission range. We also show NS-2.27's incorrect view of node 0's location in Figure 8 with a star. This incorrect view places node 0 in range of node 26, explaining the successful transmission.

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 8). 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.

3.3. Simulation Results Analysis

An entire simulation (node movement and network traffic) 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 not help a researcher trace the route of a packet. The iNSpect animation allows quick analysis of packet routes. 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 9 shows a snapshot of a Location Aided Routing (LAR) simulation [11]. LAR routing uses knowledge of

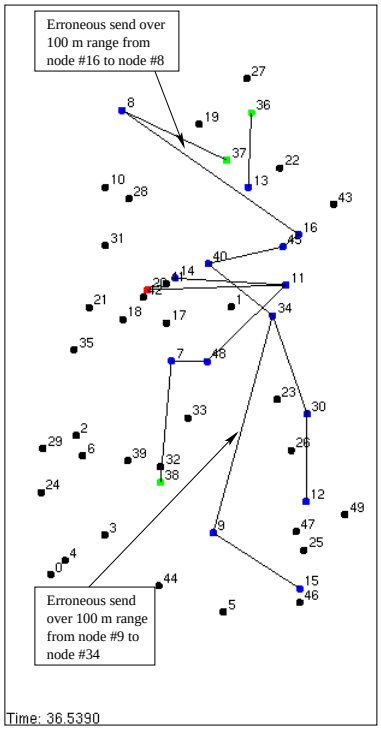


Figure 7. iNSpect showing an NS-2.27 model error. Two transmissions exceed the 100 m node transmission range. The simulation area is 300 m x 600 m.

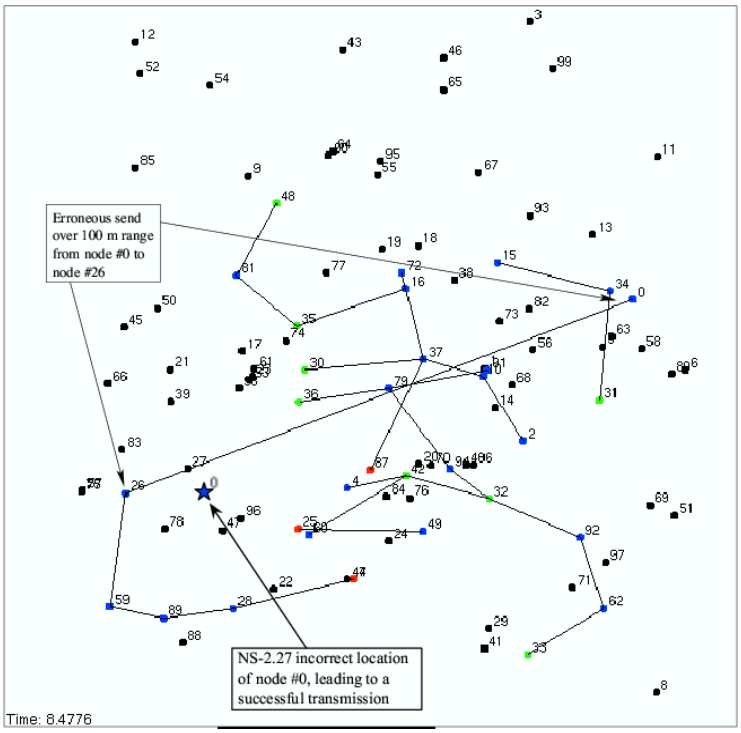


Figure 8. iNSpect showing an NS-2.27 model error. The node 0 transmission exceeds the 100 m transmission range, because NS-2's incorrect view of node 0's location places node 0 in range of node 26. The simulation area is 600 m x 600 m.

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 9 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 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 [4]. Figure 10 shows our route optimization found 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 9.

Using the suite of calculations iNSpect provides, further increases the analysis techniques available to researchers. For example, we executed the same protocol with the same scenario and produced drastically different delivery ratios. Using iNSpect's performance statistics, connectivity graph, and partition check tools, we found the scenario had a large group of nodes isolated from the rest. As a result, when the source and destination nodes were in the same 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 validate the differing performance statistics returned from the simulations.

4. iNSpect Details

In this section we discuss details of the many iNSpect features. The iNSpect program has several calculations it can make for a scenario, overlay capabilities, and other design features to provide the researcher with a powerful visualization and analysis tool.

4.1. iNSpect Calculations

Connectivity Graph - The connectivity graph tool renders a line between nodes that are within range of each other based on the transmission range of a node. The graph helps researchers in model validation and results analysis.

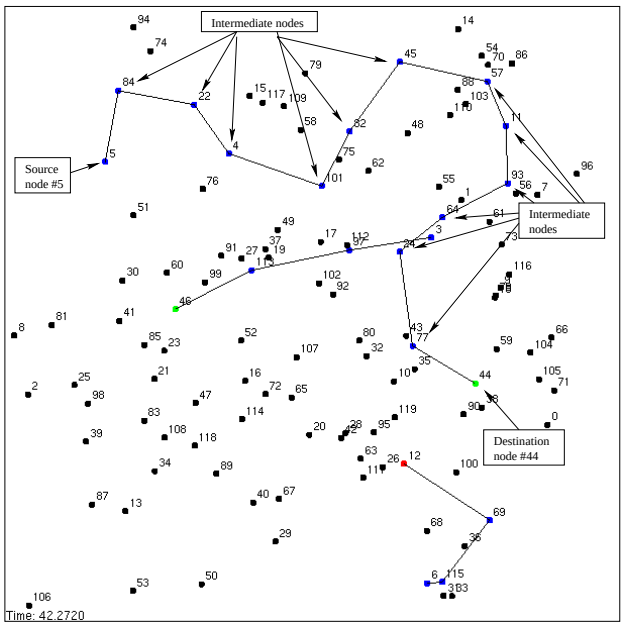


Figure 9. iNSpect showing a Location Aided Routing route selection for a transmission from node 5 to node 44 in a 600 m x 600 m simulation area with a 100 m transmission range.

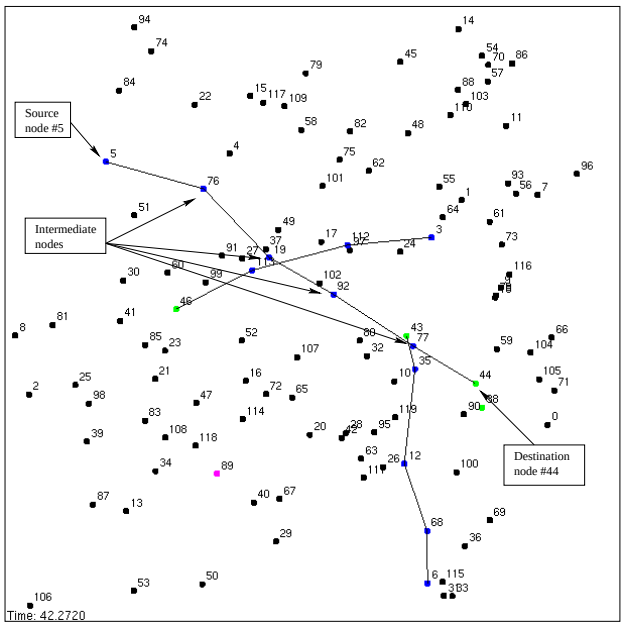


Figure 10. iNSpect showing our *projection method* [4] of Location Aided Routing route selection for a transmission from node 5 to node 44 in a 600 m x 600 m simulation area with a 100 m transmission range.

Figure 11 shows the iNSpect program with the connectivity graph illustrated. The connectivity lines can be used to determine shortest path, 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.

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 nodes 0 and 9 are partitioned at the bottom of the network area. 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 by itself. However, iNSpect is already scanning and rendering each node so visualizing partitioning is a quick calculation during the playback.

Performance Statistics - The performance statistics tool displays current statistics for each node during simulation playback. The tool calculates continuous statistics for the number of packets originated, forwarded, received, and dropped by a node. The statistics are displayed as individual histograms above each node in the network. Figure 12 shows a network with performance statistics activated. The histogram bars from left to right are number of packets sent, received, forwarded, and dropped. The iNSpect performance tool provides the researcher with continuous information per node that has never been previously available (as far as we are aware). The tool can highlight problem nodes that traditional `awk` scripts and summarizing statistics are unable to provide (see Section 3.3). For example, a researcher can identify a node that is dropping a larger number of packets than the other nodes or using a larger amount of energy.

Summary Reports - The iNSpect program generates reports of the performance statistics it calculated during the simulation. These reports can be generated anytime throughout the playback. The NS-2 trace file only contains send, receive, and some drop (collision only) packet events. The iNSpect program can link these events to calculate the number of packets forwarded, dropped, and received successfully at the destination. Both a node report and a summary report are shown in Figure 13. The *Summary Report* includes the number of packets sent, number of packets delivered, overall delivery ratio, total end-to-end delay of all packets, and average end-to-end delay. The *Summary Report* can be used in place of, or in conjunction with, post-processing scripts that generate result summaries. The *Node Report* includes the total number of packets sent, received, forwarded, dropped, and delivered for each node

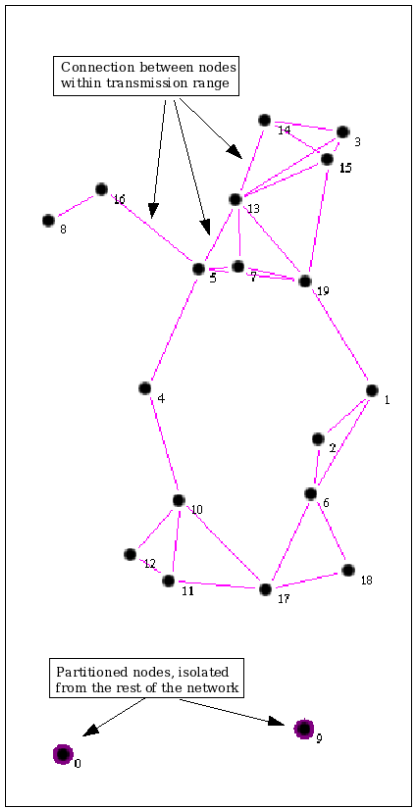


Figure 11. iNSpect display of the connectivity graph with partition check on. Nodes 0 and 9 are partitioned. 20 nodes, 300 m x 600 m simulation area.

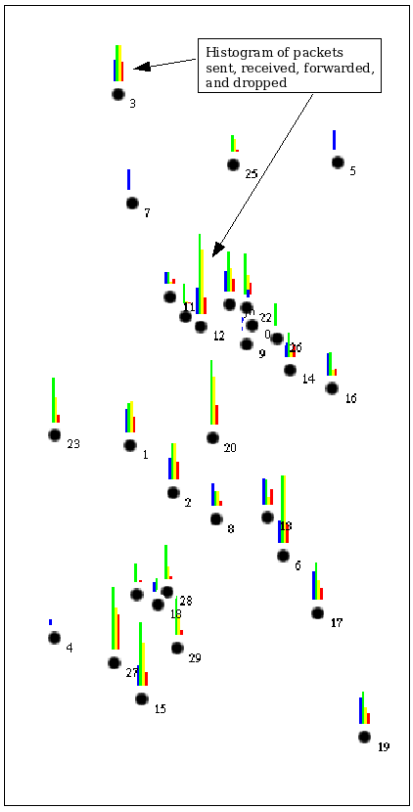
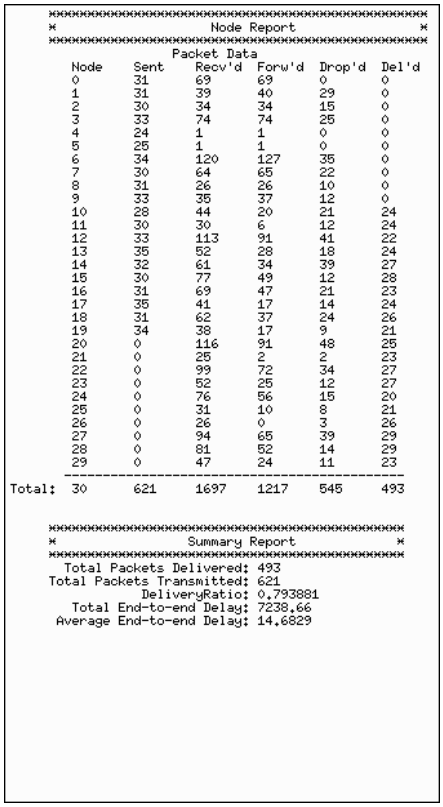


Figure 12. iNSpect display of the performance statistics for each node. 30 nodes, 300 m x 600 m simulation area.



Node Report

Packet Data

Node	Sent	Recv'd	Forw'd	Drop'd	Del'd
0	31	69	69	0	0
1	31	39	40	29	0
2	30	34	34	15	0
3	33	74	74	25	0
4	24	1	1	0	0
5	25	1	1	0	0
6	34	120	127	35	0
7	30	64	65	22	0
8	31	26	26	10	0
9	33	35	37	12	0
10	28	44	20	21	24
11	30	30	6	12	24
12	33	113	91	41	22
13	35	52	28	18	24
14	32	61	34	39	27
15	30	77	49	12	28
16	31	69	47	21	23
17	35	41	17	14	24
18	31	62	37	24	26
19	34	38	17	9	21
20	0	116	91	48	25
21	0	25	2	2	23
22	0	99	72	34	27
23	0	52	25	12	27
24	0	76	56	15	20
25	0	31	10	8	21
26	0	26	0	3	26
27	0	94	65	39	29
28	0	81	52	14	29
29	0	47	24	11	23
Total:	30	621	1697	1217	545	493

Summary Report

Total Packets Delivered: 493
Total Packets Transmitted: 621
DeliveryRatio: 0.793881
Total End-to-end Delay: 7238.66
Average End-to-end Delay: 14.6829

Figure 13. Sample Node Report and Summary Report for a 30 node simulation.

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

4.2. iNSpect Overlays

Geometric Shapes - The iNSpect program allows a user to display geometric objects, such as circles and rectangles, which may identify regions of interest. For example, Figure 14 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 [2]. (The nodes are initialized in the steady state distribution of the Random Waypoint Mobility Model [17], [18].) 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, validating 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 circle is configurable within iNSpect.

A rectangular overlay is also available in iNSpect. We use this rectangular overlay in evaluating geocast routing protocols. In geocast routing, packets are forwarded to nodes in a specific geographical area of the simulation [9]. 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 15 depicts a rectangular area of interest with corners at (150 m, 400 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 [8], 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.

Node Location - The iNSpect coordinate overlay displays node locations in (x,y) coordinates; see Figure 15 for an example. Location information can be used to evaluate

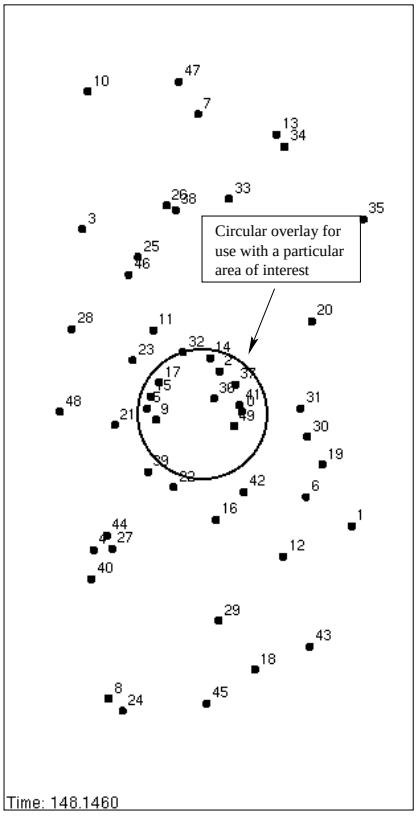


Figure 14. iNSpect display of a 30 m radius circle overlay in the center of the simulation area.

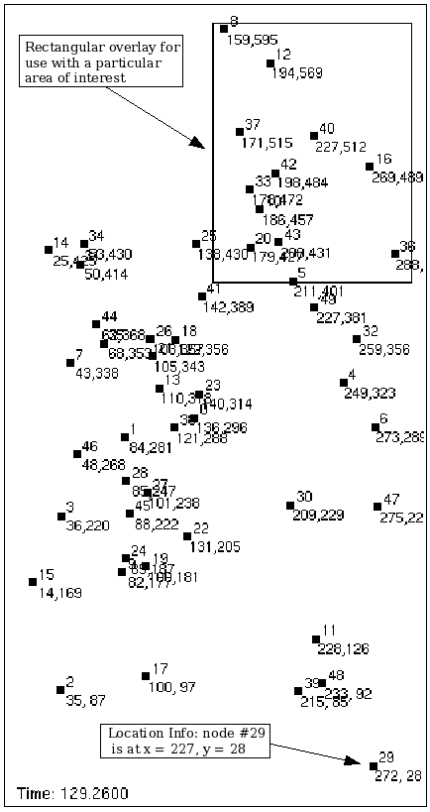


Figure 15. iNSpect display of both a 150 m x 200 m rectangle overlay and node location coordinates.

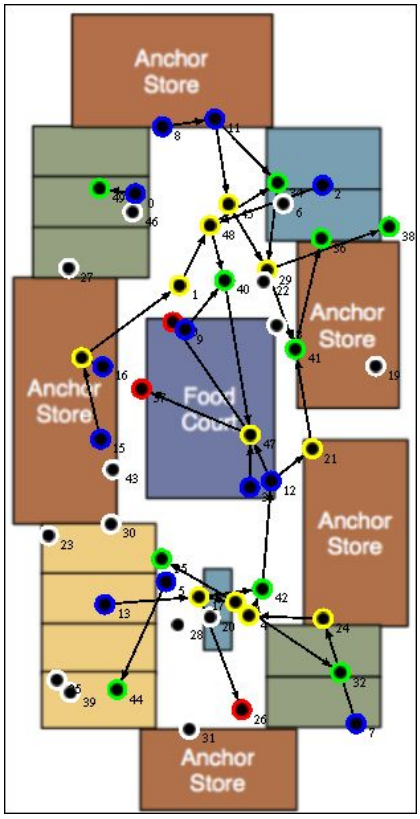


Figure 16. iNSpect display of a scenario over an image of a mall with custom node symbols.

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

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 [13]. 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.

Image Display - The iNSpect program allows the researcher to display background and node 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, another image can be loaded for display. Additionally, the individual nodes are displayed with images, which means the node images can be changed as well. Figure 16 shows an example of a scenario playback with a shopping mall map loaded as the background and larger custom node symbols. The image display capability allows a researcher to use images of laptops or cell phones for nodes on a real background. The image display adds context to the scenario for education and presentation purposes.

4.3. iNSpect Implementation

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 17. 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 back or forward 5 seconds, respectively. The Pause/Resume button works as one would expect; Figure 17 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") are discussed in Sections 4.1 and 4.2.

The Stats tab in the iNSpect GUI contains buttons for the statistics functions of iNSpect. The statistics buttons are discussed in Section 4.1. Finally, all controls can be

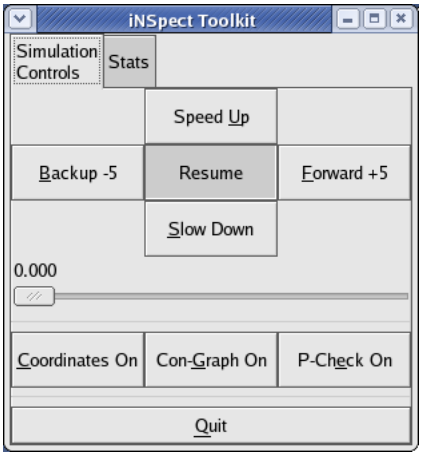


Figure 17. The iNSpect graphical user interface. Simulation controls, which control the visualization of an NS-2 simulation, are shown.

accessed using a hot key from both the toolkit and simulation windows. These hot keys are underlined in the iNSpect GUI, and are listed in Table 1. As an example, the 'p' key can be pressed once to pause the simulation and then again to resume the simulation.

Table 1. List of iNSpect Hot Keys

Key	Description
p	Toggle Pause/Resume
f	Jump forward 5 seconds
b	Jump back 5 seconds
u	Speed up
s	Slow down
c	Toggle node coordinates
g	Toggle connectivity graph
e	Toggle partition check
v	Display visual statistics
r	Generate reports
q	Quit iNSpect

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 reader is implemented as a thread to eliminate blocking between the file parser and display. This approach enables the researcher to view the NS-2 simulation scenario quickly and smoothly, and avoids a wait of several minutes to pre-process a large trace file.

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.

4.4. Additional uses

The iNSpect program can also be used to validate 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.

The iNSpect program utilizes the OpenGL [22] libraries for rendering the visualizations. 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 visualizations 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++, 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. Future Work and Conclusions

We are looking to expand iNSpect in several areas. For example, we plan to include a zoom control and the capability to make iNSpect movies for use with presentations.

Also, we plan to augment iNSpect to function with other NS-2 trace file formats and other simulators' trace files. Another area of future work concerns visual statistics. Visual statistics allow different types of analysis to compliment each other in one tool. For example, iNSpect could provide histograms of the node positions and distribution. A histogram of the number of nodes at each x-coordinate and y-coordinate helps a researcher determine whether the steady-state distribution of node positions has been reached [17]. These future capabilities will increase iNSpect's application to wireless network research.

With the increase in wireless network research, visualization and analysis of node behavior, simulations, and results are necessary to engage in productive development. The iNSpect program is a quick, low-overhead solution to animating mobility model files, topologies and simulation results 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 hours of detailed detective work trying to validate results. 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. To see iNSpect in action and obtain our iNSpect code go to `http://toilers.mines.edu/iNSpect`.

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/visualization trace file.

References

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

[2] 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.

[3] 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 April 14, 2005.

[4] 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.

[5] The GNU image manipulation program GIMP toolkit. http://www.gtk.org. Page accessed on April 18, 2005.

[6] J. Grover. How to draw graphs from NS trace files. http://mailman.isi.edu/pipermail/ns-users/2004-May/041965.html. Page accessed on April 14, 2005.

[7] A. Gurtov and S. Floyd. Modeling wireless links for transport protocols. *ACM Computer Communication Review*, 34(2):85–96, 2004.

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

[9] 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.

[10] H. C. Kee. NAM support for wireless traffic. http://mailman.isi.edu/pipermail/ns-users/2004-May/042046.html. Page accessed on April 14, 2005.

[11] Y. Ko and N. 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.

[12] S. Kurkowski, T. Camp, and M. Colagrosso. MANET simulation studies: The current state and new simulation tools. Technical report, Department of Math. and Computer Sciences, Colorado School of Mines, MCS-05-02, February 2005.

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

[14] J. Mehringer. The NAM editor. A presentation for the CONSER retreat, slide 3, 2001. URL: http://www.isi.edu/nsnam/nam/nam-editor.ps. Page accessed on April 15, 2005.

[15] Monarch Project: Ad-hockey for Perl/Tk800.015. http://www.monarch.cs.rice.edu/cmu-ns.html. Page accessed on April 14, 2005.

[16] Monarch Project: Wireless and mobility extensions to NS-2. http://www.monarch.cs.rice.edu/cmu-ns.html. Page accessed on April 1, 2005.

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

[18] 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'04)*, pages 319–326, 2004.

[19] The network simulator - NS-2. http://www.isi.edu/nsnam/ns/. Page accessed on April 1, 2005.

[20] 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.

[21] K. To. Bug in ns-2.27 wireless channel. http://mailman.isi.edu/pipermail/ns-users/2004-April/041388.html Page accessed on April 1, 2005.

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