Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

Research and Analysis of Simulation-based Networks through Multi-Objective Visualization[1]

J. Mark Belue
Stuart H. Kurkowski
Scott R. Graham
Kenneth M. Hopkinson
Ryan W. Thomas
Joshua W. Abernathy

Department of Electrical and Computer Engineering
Air Force Institute of Technology 2950 Hobson Way
Wright-Patterson AFB, OH 45433, U.S.A.

ABSTRACT

Visualization of individual network events is a crucial part of testing new network designs and analyzing network performance and efficiency. This research designed and developed a framework for visualizing complex military and non-military wired and wireless networks. Our framework provides a robust network simulator trace file parser, multiple network visualization layouts–including user-defined layouts, and precise visualization controls. The parser architecture is capable of accepting trace files from different network simulators and provides one common visualization environment to study network scenarios run on different simulators. The many dynamic multi-objective network views add to the analyst's suite of tools available for analyzing networks. Analysts can toggle between the different views to provide even greater analysis capability. We describe our methodologies for the design and provide example analysis scenarios. Our framework will allow researchers to advance the state of network simulation-based analysis.

1 INTRODUCTION

The World Wide Web enables people from all over the world to view web pages filled with a wide range of content with high availability and low delay. The phenomenal and seemingly ever-present capability, built upon the fixed infrastructure of the Internet, has fostered tremendous innovation in the way business is conducted. A side effect is that it has created a voracious human appetite for data and connectivity and impatience with systems which fail to provide it. A natural result of this capability is that military leaders desire the equivalent capability for military operations. Indeed, Net-Centric Warfare is repeatedly touted as key to continuing US military superiority.

For military networks, mobility exists not just at the edge of the network, but at the core, where aircraft, ships, and trucks would be the equivalent of mobile cell-towers. This presents many significant problems, from a mixture of wired and wireless links, to a majority of wireless links, to dynamic topologies—each of which causes significant disruptions in connectivity. The forward or deployed military network is a harsh environment. While messages are rarely corrupted in wires and fiber optic cables, bandwidth can be limited; wireless communications must employ a vast array of error correcting schemes for successful connections. Additionally, as mobile nodes move around, the wireless channel changes frequently. Nodes may move in and out of range, resulting in persistently intermittent connections. Dynamic and adaptive protocols, devices, and operational strategies must be researched and analyzed.

Research to address these areas is done primarily with simulation. Using simulation, network architects can design a network inside a network simulator and then analyze through summary statistics, how the network will perform under specific scenarios. Visualization of individual network events is a crucial part of testing new network designs and analyzing network efficiency. Visualization quickly reveals sources of network inefficiencies such as: queuing delays, dropped packets, network congestion, and insufficient bandwidth. Additionally, network visualizations provide protocol developers a powerful method to quickly evaluate their protocols by visually analyzing network packet traffic.

Our research provides a visualization framework, that enables researchers to render multiple layouts of the network events and activities highlighting different aspects of their scenarios. We provide at least three major contributions with this new framework. First, a trace file parsing architecture that is easily extensible to include traces in numerous formats. We have currently tested it on NS-2 and OPNET simulation trace files. Second, a multi-objective network topology visualization layout that can be easily extended to highlight specific user network behavior interests. Finally, an analysis-based user interface to control the visualizations.

Our effort seeks to advance network analysis and research with a framework designed to allow visualization customization for network topology and network events. Our toolkit is currently being used for Air Force network planning, design, and analysis.

This paper is made up of the following areas. We will discuss other network visualizations and toolkits in Section 2. In Sections 3 we discuss our design methodology and techniques. In Section 4 we detail our multi-objective dynamic layouts. Then in Section 5 we present two network simulator scenarios and a validation to show how multi-objective layouts increases researcher awareness and understanding. Finally Section 6 provides some conclusions and future work.

2 NETWORK SIMULATION & VISUALIZATION

Simulating network events through network simulation is a well established method used in network research endeavors to understand and test network events, network traffic and protocols. According to (Kurkowski, Camp, and Colagrosso 2005a) over 75.5% of mobile ad hoc Network research is done with simulation. In addition to network event simulations network event visualizations provide insight and understanding of simulated network events. Combining simulation with visualization provides an effective and efficient method for network research and analysis.

The existence of several different visualizers built from the ground up for each simulator raises the questions: Can a single network visualization framework be created to effectively visualize both wired and wireless network simulator scenarios? What about network scenarios created in different simulators (NS-2, OPNET, etc.)? This framework combines network simulator trace data with a well-established visualization toolkit to create an efficient and robust framework. This new visualization framework accurately animates network events contained in a simulation trace file.

2.1 Network Simulator Visualizations

A short review of existing network simulator visualizers motivates the creation of a new visualization framework. We reviewed the Network Animator (Nam) (Estrin et al. 2000), interactive NS-2 protocol and environment confirmation tool (iNSpect) (Kurkowski, Camp, and Colagrosso 2005b), Georgia Tech Network Simulator (GTNets) (Georgia Institute of Technology 2008) and the Optimized Network Evaluation Tools (OPNET) (OPNET Tech 2008). The strengths and weaknesses of these visualizers provided the basics requirements for the development of a new network visualization framework.

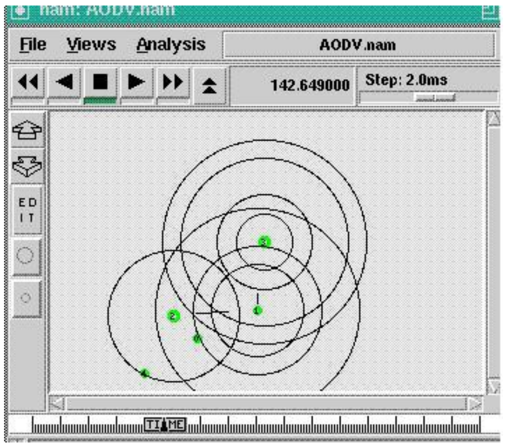


Figure 1: Nam visualization showing node broadcast rings.

When NS-2 executes a network scenario it produces a trace file, containing the timed stamp network events that occurred during the scenario. The general practice among researchers is to use the trace file to calculate summary statistics and scenario averages. Both Nam and iNSpect are network visualizers that playback NS-2 trace files. Nam provides packet-level animation, protocol graphs, traditional time-event plots of protocol actions and scenario editing capabilities (Estrin et al. 2000). Additionally, Nam allows user interaction with the simulator through the scenario input facility, but is limited to setting up network scenarios prior to execution. The scenario input facility uses traditional drawing approaches to add nodes, links, and protocol agents (Estrin et al. 2000).

Originally built to support wired network visualization Nam has limited wireless network animation capabilities. Nam visualizes wireless traffic as broadcast rings moving outward originating from the center of wireless nodes (see Figure 1). The rings make wireless traffic difficult to track during a network scenario.

The iNSpect program visualizes wireless network events such as packet hops, wireless node links, and packet delivery success (Kurkowski, Camp, and Colagrosso 2005b). The iNSpect program uses colored arrows to better highlight wireless routes and eliminate confusion, but iNSpect only visualizes wireless traffic. These drawbacks of Nam and the iNSpect program motivate the need for a new and robust network visualization for both wired and wireless networks.

Unlike Nam and iNSpect, GTNets combines a full-featured network simulation environment with graphical viewing of the simulation. The network simulation suite shows the network topology along with network traffic. GTNets also provides fine-grain control mechanisms, but the visualizer is tightly coupled to the simulator as the means of running the simulation (Georgia Institute of Technology 2008). As a result, the visualizer cannot be separated from the simulator nor can it work with other simulators' outputs.

Like GTNets, OPNET (OPNET Tech 2008) combines a network event simulator with a network event visualization for network packet events and node movement. The visualization includes node broadcast rings, node movements, and animated packets moving across wired and wireless links. Although OPNET is a powerful network simulator suite it's proprietary implementation is significantly difficult to alter. Like GTNets, the visualizer cannot be separated from the simulator.

Drawing on the successes and shortcomings of NAM, iNSpect, GTNets, and OPNET we developed a new network visualization framework that visualizes both wired and wireless network events including: traditional packet animation across wired and wireless links, queue levels at each node, dropped packets, node movement, and playback controls. Additionally, the framework will run independent of a network simulator and handles multiple trace file formats. By visualizing both wired and wireless networks this framework provides a single source tool that can be used to research and analyze network simulation scenarios.

2.2 Visualization Toolkit

The open source ***prefuse*** library is an extensible software toolkit for helping software developers create interactive information visualization applications using the Java programming language. The ***prefuse*** library simplifies the processes of data handling, representation, and mapping to on-screen displays as well as crafting direct manipulation interactions with the visualization (Heer 2008).

The ***prefuse*** toolkit provides many visualization tools that are integrated into our framework design. Using the ***prefuse*** toolkit for network visualization provides features such as panning, zooming, and layout control of visualization objects. Additionally, integration of these features into our framework facilitates effective visualization development and lead directly to our layout aided analysis capabilities.

The ***prefuse*** library has four parts to its architecture: data, filtering, rendering and views, and interaction. The ***prefuse*** data and filtering is accomplished through a trace file parser, which parses data into the ***prefuse*** visual forms. The ***prefuse*** rendering and view handles the animation of the network events.

3 OUR FRAMEWORK DESIGN

Behind the visualizations, the three major design challenges for this framework were: network simulator trace file parsing, wired and wireless network event animation, and analyst interaction.

3.1 Network Trace File Parser

The network visualization framework requires a trace file produced by the network simulator to visualize network events. Because NS-2 directly produces trace files with the needed network event information this research started with parsing NS-2 trace files.

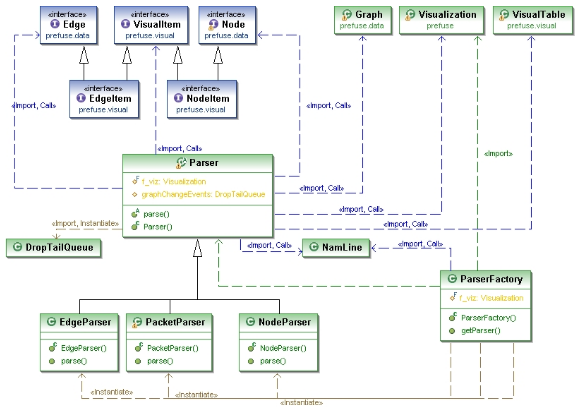


Figure 2: Parser object design.

The polymorphic parser design creates one consolidated location for specifying parser criteria, making parsing changes simple and straightforward (see Figure 2). In addition to the `ParserFactory` the concrete products are limited to only node, edge, and packet parser objects. Finally, the `Parser` abstract class handles updates to the visualization through the `node`, `edge`, and `packet` parser objects. This design establishes `Parser` as the central manager for updates to different objects held in the ***prefuse*** architecture. We show the design's extensibility with a validation using OPNET in subsection 5.3.

3.2 Network Event Animation

Our visualization framework handles the control and processing required to maintain accurate scenario playback and analysis. The framework does this using a `netviz_animation` thread to parse the trace file into the `Visualization` object and update the layout of the `VisualItem` objects. Meanwhile the `netviz_simulationClock` keeps track of the virtual clock in the visualization. This clock is displayed on screen during the visualization and is synchronized with network events and framework calculations discussed below (i.e., node movement, etc.). Our framework implementation displays and animates ***prefuse*** `VisualItem` data types to visualize both wired and wireless networks in the same framework, overcoming one of the limitations Nam and iNSpect. Figure 3 shows the visualization interface, with a 10-node military scenario.

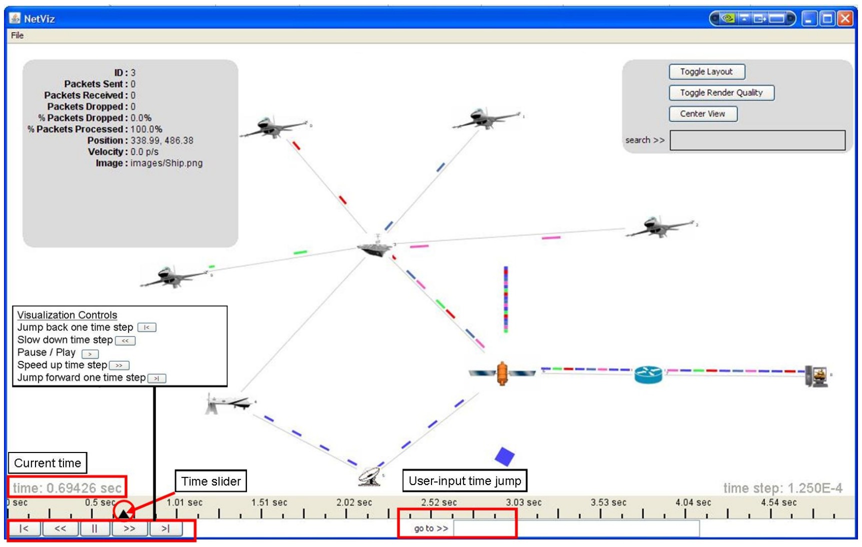

ID : 3
Packets Sent : 0
Packets Received : 0
Packets Dropped : 0
% Packets Dropped : 0.0%
% Packets Processed : 100.0%
Position : 338.99, 486.38
Velocity : 0.0 p/s
Image : images/Ship.png

Figure 3: Screen shot showing fighter aircraft, an unmanned aerial vehicle, ground station, satellite, and a ship.

To provide an accurate interactive research and analysis visualizer for network simulation, we need to represent numerous elements of a simulated network scenario. These elements include wired and wireless nodes, their movement, and other characteristics (i.e., buffers); both wired and wireless links between nodes; and packet animation to include queues and transmission. Additionally, the analyst needs to interact with each of these various elements.

3.2.1 Wired and Wireless Nodes

Nodes and their depiction are a key aspect of network visualization. In our framework researchers can use graphical shapes as well as images to represent nodes. The default is a circular node. Images, however, provide greater context and awareness for research and analysis as well as presentation.

Increased use of mobile nodes in networks warrant the need to accurately show mobile node movement. Our framework uses the node movement information given in the trace file using the defined tags '-u' for x-coordinate velocity, '-v' for y-coordinate velocity, '-T' for duration of movement (Fall and Varadhan 2002). These lines only appear when movement changes occur for each node, so our visualization framework must calculate these node movements, corresponding to the x and y velocity and duration. Movements must be scaled to accommodated zooming, panning, as well as the playback speed selected by the analyst.

Another characteristic of nodes in a network is their buffers. Our framework calculates buffer levels and displays that graphically for each node (see Figure 3). Additionally, as packets are dropped from the queue, our framework displays the dropping packets.

The interaction for a node includes focus (magnification of the shape or image) when the mouse is moved over the node. A node can also be selected, displaying amplifying information in the corner of the display. This information currently includes statistics for packets sent, received, and dropped, node position, velocity, and image name (see the upper left corner of Figure 3).

3.2.2 Network Links

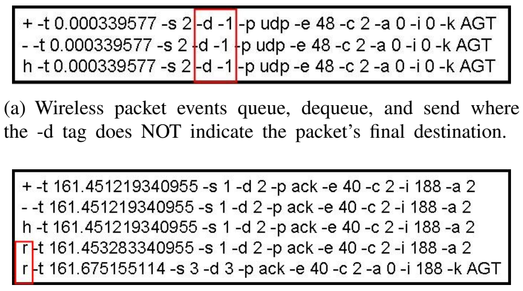
Communication networks are formed based on links between nodes. For wired and directional antenna-based wireless networks this requires the display of links between network nodes. Graphically these links are shown as lines on the display. Our framework creates the lines based on trace file information. However, as link status changes and nodes move, the rendering of the link lines must be maintained. Our framework calculates the line placement to match the location of the nodes in the renderer's coordinate system.

Visualization of omnidirectional wireless links are different from the wired links, because they are not explicitly listed in the trace file. Wireless packet transmissions (from nodes with omnidirectional antennas) are not constrained to a graph-based connection. Wireless packets are broadcast and can be received by many different nodes. Nodes that receive the broadcast can ignore it, process the packet, or relay the packet to surrounding nodes. Visualizations of transmission energy rings disseminating from a node, although realistic, quickly become confusing and do not highlight a packet's route from source to destination (see Figure 1). Additionally, the nature of wireless broadcasts does not allow a packet's path to the destination to be known at the time of initial packet broadcast from the trace file. A wireless link is not established until the packet is received at a node. At this point the link can be rendered and is very useful in protocol route tracing (Kurkowski, Camp, and Colagrosso 2005b). However, it must be determined on a per packet basis, see subsubsection 3.2.3.

Like nodes, links have user interactions as well. An analyst can select a link and get amplifying information displayed about the link, such as bandwidth, etc.

3.2.3 Packet Transmission

In our framework packets are displayed on the links as they are transmitted between nodes. The packets are drawn proportional to their size and the bandwidth of the link. The packet timing is also accurate from sender to receiver. To do this the framework calculates the distance and timing to get accurate scaling. Additionally, our framework tracks the packets to color code them based on packet flows. This tracking continues from source to destination and all nodes and queues in between. This explains the multi-colored packet buffers in some of the figures in this paper.



```
+ -t 0.000339577 -s 2 -d -1 -p udp -e 48 -c 2 -a 0 -i 0 -k AGT
- -t 0.000339577 -s 2 -d -1 -p udp -e 48 -c 2 -a 0 -i 0 -k AGT
h -t 0.000339577 -s 2 -d -1 -p udp -e 48 -c 2 -a 0 -i 0 -k AGT
```

(a) Wireless packet events queue, dequeue, and send where the -d tag does NOT indicate the packet's final destination.

```
+ -t 161.451219340955 -s 1 -d 2 -p ack -e 40 -c 2 -i 188 -a 2
- -t 161.451219340955 -s 1 -d 2 -p ack -e 40 -c 2 -i 188 -a 2
h -t 161.451219340955 -s 1 -d 2 -p ack -e 40 -c 2 -i 188 -a 2
r -t 161.453283340955 -s 1 -d 2 -p ack -e 40 -c 2 -i 188 -a 2
r -t 161.675155114 -s 3 -d 3 -p ack -e 40 -c 2 -a 0 -i 188 -k AGT
```

(b) Packet received at node two, then received at node three without wireless broadcast.

Figure 4: Wireless traces of four unique packetIDs.

However, rendering these packets is not a trivial reading of the trace file. A trace file send event indicates the node and the time a packet is sent. For linked nodes, the packet can be sent and displayed on the link. However, the information needed to visualize wireless packets moving from node to node is not contained in a single event trace. For example broadcast events do not show the individual recipients, as shown in Figure 4. Figure 4(a) shows a wireless event ('h' for hop) from node two. The destination ('-d') is '-1' meaning no specific destination, but a broadcast. Therefore, a link cannot be established at this point. Figure 4(b) illustrates packet traffic moving from wired links to a wireless link. The first receive ('r') clearly shows the source as one ('-s 1') and the destination as two ('-d 2'). However, the second receive ('r') is at node three without a previous send event. Because of the ambiguous nature of NS-2 trace files for wireless packet events, our framework must pre-processes trace file data by reading ahead. Reading ahead enables events which create packet broadcasts to be linked to a corresponding packet receive event. This allows us to correctly render the packet trajectory and scale the packet size, a key for accurate network analysis.

Like nodes and links, packets have user interactions as well. An analyst can select a packet and get amplifying information displayed about the packet identification number and the flow, etc.

3.3 Discussion

The network visualization framework discussed above creates an accurate improved way to study wired and wireless network nodes, links, and packet activity. Research on network protocols and network design will be aided by the ability to see wired and wireless network traffic and communicate with network simulators. Additionally, because this framework is built on the ***prefuse*** visualization toolkit new file parsing formats can be added to accommodate other simulators independently. The network visualization framework provides new visualization capabilities and a solid basis for future development, overcoming many limitations of Nam, iNSpect, GTnets, and OPNET.

4 NETWORK DISPLAY METHODOLOGIES

The previous section detailed the nodes, links, packet animation, and interactions. Here we explore the visual layout of the network entities. As networks get more complex, the question needs to be asked, are there different ways to visualize a network? We think there are and we discuss physical as well as other graph-based layouts in this section.

4.1 Physical Network Layout

When implementing a network visualization it is usually a mechanical concern when it comes to visualizing the network topology. The obvious layout is a Cartesian coordinate-based physical rendering. As a result, the framework merely calculates the x,y, z plot and renders the entities. For our physical representation of the network, we do just that. The location of each node is provided in the trace file "node" event lines as described previously. Our framework then translates from the simulator's x,y,z coordinate system to the display's x,y,z coordinate system. The translation maintains proportionality and context. For example, if a wireless node has a certain transmission range and is communicating with a particular node at the edge of the range, that should be displayed proportional to a communication activity of a node inside that range. Our physical layout provides the researcher precise user interface controls including: zooming, panning, node highlighting, and visualization timing controls.

4.2 Graph-based Network Layout

Sometimes a physical layout does not provide sufficient information about how the network is working or the network protocol/problem being studied. Nodes can appear on top of each other and packets can quickly become congested and blur together. For these reasons, a physical network layout is not always best for analyzing and observing network events and a more flexible graph-based layout is needed. This capability employed on a real network would greatly increase network situational awareness.

Our framework provides the capability to easily alter the visualized network layout to analyze different objectives in a scenario. We currently have two categories of these multi-objective dynamic displays. The first is a grouped layout and the second is a force-directed layout.

4.2.1 Grouped Layout

The grouped layout releases the nodes from their physical x,y,z coordinate-based locations and allows the nodes to be grouped in virtual clusters. The default behavior of the framework is to cluster nodes based on communication. So, as a node exchanges packets with another node, the nodes will be moved together based on links that exists between the nodes. The results is these links or more specifically characteristics of these links contribute to the rendered visualization. The groups are established by identifying the nodes that initiate activity and then determine the nodes which are recipients of that activity. The cohesion of the group is then applied to the nodes as gravity, so the nodes move together smoothly.

4.2.2 Force-directed Layout

The ***prefuse*** library implements a force-directed visual item that can allow nodes to be primarily free floating, but constrained by gravitational forces. The ***prefuse*** library uses the fourth-order Runge-Kutta integration algorithm (Cartwright and Piro 1992) to calculate the forces of the spring and drag coefficients as well as the spring length four times per time step. The actual implementation in ***prefuse*** uses the (Barnes and Hut 1986) method.

The force-directed capability can be applied to any number of network characteristics. Characteristics applied to the spring coefficients and length influence how attracted the nodes are to each other in both a positive or negative way. For example, if the spring coefficient was $\kappa_s = D_{ete}$, where κ_s is the spring coefficient and D_{ete} is the end-to-end delay, then the links with lower end-to-end delay have less spring. As a result nodes that share lower end-to-end delayed links are pulled closer together and nodes that share higher end-to-end delayed links appear farther apart.

The drag coefficient effects the constraints on the overall node movement. If the drag is low the nodes tend to move constantly and spring each other around the display. Adjustments to the drag and spring coefficients as well as the average spring length have been made in the framework to smooth out the movements.

4.3 Discussion

These dynamic views add to the analysts suite of tools available for researching network performance. Additionally, analysts can toggle between the physical and grouped layout views to provide even greater analysis capability. This multi-objective approach of the physical layout toggled with the activity-grouped layout can quickly highlight packet destinations and network activity, better than summary scenario statistics like average end-to-end delay, etc.

5 APPLICATION AND VALIDATION

In this section we contrast the physical network layout with the grouped and force-directed layouts of real scenarios to highlight the benefit of these dynamic layouts. We also discuss a brief validation of our frameworks robust file parser design, by added an OPNET parser capability.

5.1 Application I: Wireless Network Broadcast

Effectively visualizing network events requires the visualization logic to create a picture where users can quickly and accurately understand what is happening in the network. Wireless networking is a difficult venue for network analysis. As discussed earlier, wireless packets are often broadcast to multiple nodes and without link lines highlighting a packet's path it is difficult to know a packet's final destination. Tracing wireless packet routes becomes even more difficult as more and more wireless nodes broadcast and the areas around nodes becomes more and more congested with packet traffic.

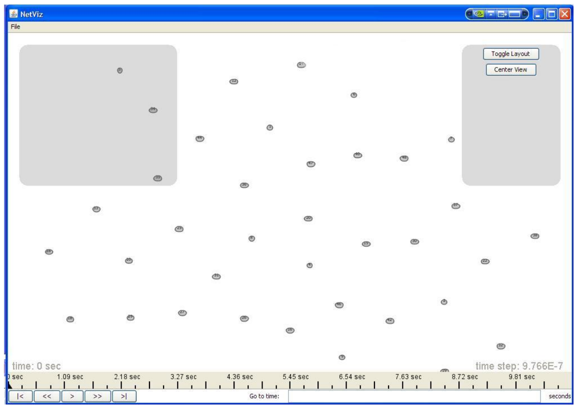


Figure 5: 50 wireless nodes prior to any broadcasts.

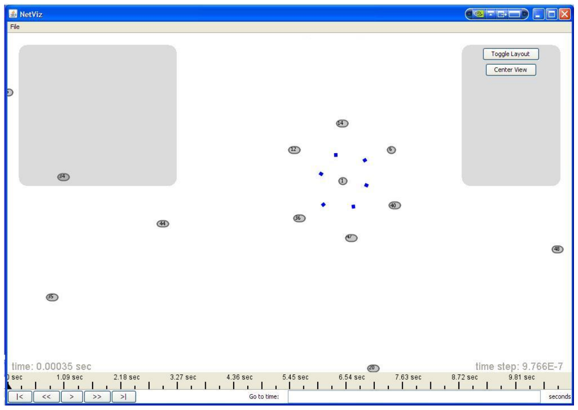


Figure 6: Packet broadcast from node 1. The grouped layout causes nodes receiving packets to move closer to node 1.

Our framework overcomes these challenges by using a grouped layout. Using grouped layout a network of many nodes will render groups based on node communication not physical x, y, z location. Nodes that broadcast to one another group together to make packet destinations obvious. For example, Figure 5 shows 50 wireless nodes in the scenario-based layout. If one of those nodes wants to broadcast a packet this single send event can generate six receive events at different nodes. In the physical layout packet animation of small colored boxes 'floating' between nodes, although accurate, can make analysis difficult, especially if the nodes were far apart or packet traffic was high. Using the grouped layout in Figure 6, the six receive events can be highlighted rapidly, because the framework groups the nodes based on network activity at the time of the event.

This interaction with multiple contexts and linked focus can aid discovery of network conditions. Network anomalies may become apparent such as: defective nodes,compromised nodes, and nodes that are obstructed from sent packet transmissions. Ultimately providing explanations for particular scenario performance observations.

5.2 Application II: Network Link Break

The force-direct graph layout defines network nodes that repel one another and network links that pull nodes together. In this example scenario, we have used current bandwidth utilization as the spring coefficient. You can also get a similar rendering by using link delay or some other link characteristic. The physical layout of this example is shown in Figure 3. In this example, the link between the Unmanned Aerial Vehicle (UAV) and the uplink site (satellite dish image) will go down. In a complex network, a link break can be very difficult to detect visually. For example, if this link went down in Figure 3 (bottom left-hand link), the analyst would have to notice packets not flowing, which might be impossible at worse or take a long time at best, with a complex network. Figure 7 shows the same scenario, with the view toggled to the force-directed graph layout. In this case, the UAV, the uplink site and the satellite are drawn closer together, based on current bandwidth available on their links. They have large bandwidth utilization, so a correspondingly large attraction between the nodes. Figure 8 shows the same scenario a little later in time right after the link between the UAV and the uplink site went down. Notice, the bandwidth between the nodes went to zero and so the spring coefficient also went down and the nodes moved significantly apart in a spring like motion. In Figure 8 the UAV is now in the upper right-hand corner of the display. Because the layout radically changed at the moment the link went down, the analyst can quickly identify the change and note the time of the event or even pause the playback. From here the analyst can select the nodes and packets or links to review amplifying information to research and analyze the problem. With the rapid identification of the event, the researcher can note the exact time. The analyst can even restart the visualization at that time in a future session, by using the framework's "user-input time jump" to move around in a long scenario. The analyst can also toggle the view back to the physical layout, providing more context to the situation.

5.3 Validation

The goals of this research were to develop a framework that enables researchers to analyze complex dynamic networks independent of a particular simulator. Secondly we wanted to provide visualizations that further the research of complex networks and aid in network analysis of simulated networks. This section discusses the validation of these two goals.

5.3.1 Framework Validation

To validate the parser design, we used an OPNET toolkit by (Coyne 2008) to generate an OPNET trace file. For OPNET the `TraceLine` abstract class follows the framework's parsing criteria explained earlier. However, trace tags used in the OPNET trace are different from those used in the NS-2 parser. Despite these differences the framework parser was easily adapted with minor extensions to parse the OPNET trace file. With the extended parser, the data was processed and displayed. The OPNET scenario was validated against the OPNET visualizer to validate that it displayed the scenario accurately.

As a result, we have now extended `TraceLine` for two different network event line traces. `NamLine` extends `TraceLine` to parse network event lines from NS-2 trace files. `OpNetLine` extends `TraceLine` to parse network events recorded from a scenario run in OPNET.

By using conditional logic based on the file type the correct `TraceLine` subclass is instantiated and additional

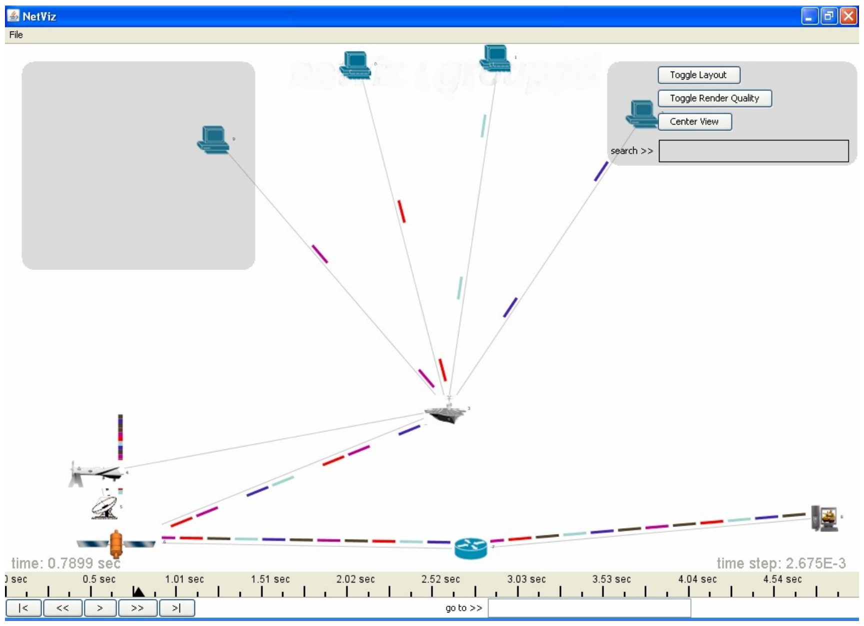


Figure 7: Wired network topology–arrows point out network links that will go down

subclasses can be added later as necessary. Once the correct `TraceLine` subclass is instantiated the developer can add specific parsing logic as needed by the new subclass in order to parse the simulator trace line.

5.3.2 General Visualization Validation

Visualizations, by their very nature, are difficult to test. This research tested numerous visualization scenarios by visual inspection. The testing was done by visualizing a network scenario, or trace file, in Nam and comparing the resultant visualization with the same trace file visualized by this framework. The network scenario trace files used during framework testing demonstrate important network event visualization capabilities.

6 CONCLUSION

Our research studied several current network visualizers to determine how their positive attributes could be incorporated into a more robust and effective network visualizer. As a result we have created a network visualization framework using the ***prefuse*** toolkit. Our framework provides a single network visualizer that runs simulator independent. It provides seamless wired and wireless visualization with various layout options.

The dynamic multi-objective layouts provide an excellent foundation for future research and collaboration into other ways to visualize networks. The framework brings network simulator research one step closer to comparing executed network simulator scenarios in a common visualization environment. A comparison that would provide better understanding on how simulators differ and how those differences can effect network event simulation. Additionally, network simulator performance could be seamlessly analyzed and compared from one simulator to another.

Future work will build upon the parser architecture to handle different network trace formats. Additionally, the flexible layout architecture will allow developers to customize the visualization to commander preferences. The framework provides network event visualization with the flexibility needed to visualize today's large and complex military networks.

DISCLAIMER

The views expressed in this document are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

REFERENCES

Barnes, J., and P. Hut. 1986. A hierarchical O(N log N) force calculation algorithm. *Nature* 324 (1): 446–449.

Cartwright, J., and O. Piro. 1992. The dynamics of Runge-Kutta method. *International Journal of Bifurcation and Chaos* 3 (2): 427–449.

Coyne, M. E. 2008, March. Hot swapping protocol implementations in the OPNET modeler development environment. Master's thesis, AFIT.

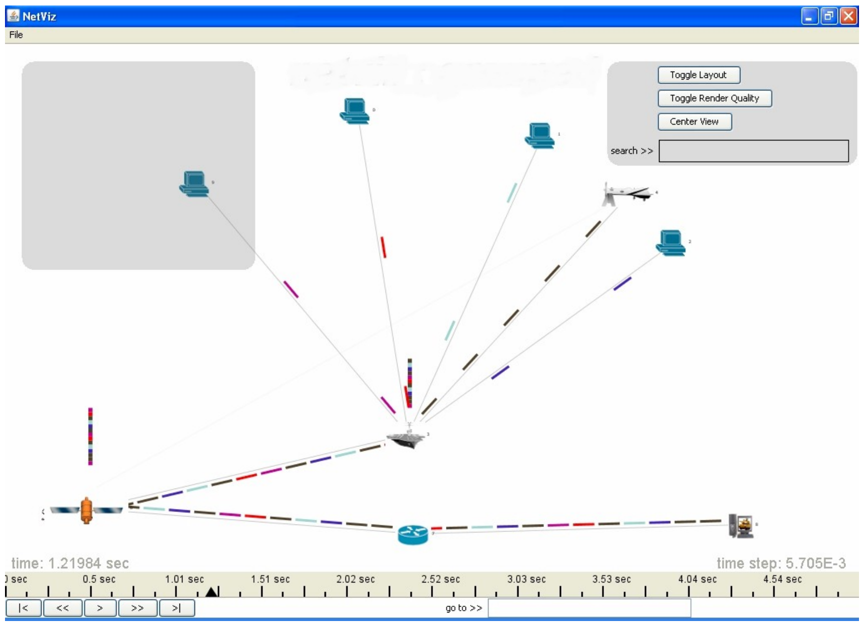


Figure 8: Altered network topology–arrow points to link that will go down and oval shows links that will go up

Estrin, D., M. Handley, J. Heidemann, S. McCanne, Y. Xu, and H. Yu. 2000. Network visualization with Nam, the VINT network animator. *Computer* 33 (11): 63–68.

Fall, K., and K. Varadhan. 2002. The ns manual.

Georgia Institute of Technology Feb. 5, 2008. GTNets. www.ece.gatech.edu/research/labs/MANIACS/GTNetS/.

Heer, J. February 5, 2008.. prefuse. www.prefffuse.org/.

Kurkowski, S., T. Camp, and M. Colagrosso. 2005b. A visualization and animation tool for ns-2 wireless simulations: iNSpect. *Proceedings of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*:503–506.

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

OPNET Tech Feb 5, 2008. Opnet. www.OPNET.com/.

AUTHOR BIOGRAPHIES

J. MARK BELUE is a Master's graduate of the Department of Electrical and Computer Engineering at AFIT, with a research focus in software engineering, and simulation. His email address is <`john.belue@afit.edu`>.

STUART H. KURKOWSKI received his Ph.D. in Mathematical and Computer Science from Colorado School of Mines in 2006. He is currently an Assistant Professor of Computer Science at AFIT specializing in software engineering, visualization, and simulation. His email address is <`stuart.kurkowski@afit.edu`>.

SCOTT R. GRAHAM received his Ph.D. in Electrical Engineering from the University of Illinois in 2004. He is an Assistant Professor of Computer Engineering at AFIT. His interests lie in the areas of networking and control systems. His email address is <`scott.graham@afit.edu`>.

KENNETH M. HOPKINSON received his Ph.D. in Computer Science from Cornell University in 2004. He is currently an Assistant Professor of Computer Science at AFIT. His research interests include fault-tolerant and distributed systems, networking, and simulation. His email address is <`kenneth.hopkinson@afit.edu`>.

RYAN W. THOMAS received his Ph.D. in Computer Engineering from Virginia Tech in 2007. He is currently an Assistant Professor of Computer Engineering at AFIT specializing in cognitive networks and wireless communication systems. His email address is <`ryan.thomas@afit.edu`>.

JOSHUA W. ABERNATHY is an undergraduate student in the Department of Computer Science at Cedarville University working as a research assistant at AFIT. His email address is <`josh.abernathy@afit.edu`>.