Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

A MANET SIMULATION TOOL TO STUDY ALGORITHMS FOR GENERATING PROPAGATION MAPS

Scott L. Rosen
John A. Stine
William J. Weiland

The MITRE Corporation
McLean, VA 22102, U.S.A.

ABSTRACT

In this paper we describe a new simulation tool used to study the creation and optimization of propagation maps for Node State Routing protocols within wireless and mobile ad hoc networks. The simulation is developed in MATLAB and interfaces with DLL libraries for network data management support. These DLL libraries also have applications within OPNET for other Node State Routing studies.

1 INTRODUCTION

Propagation maps are used in the mobile ad hoc networking (MANET) routing protocol Node State Routing (NSR) to capture how well a node can hear its neighbors. Our simulation tool, Single Hop Propagation Map Evaluation Tool (SHOPMET), supports the study of the optimization of propagation map size and the longevity of propagation map relevance.

We start this paper by describing ad hoc networks and our motivation for using propagation maps. We then describe the constituent problems in creating propagation maps before proceeding into a discussion of the simulation tool used for our study on propagation maps.

1.1 Background - What are Ad Hoc Networks

Mobile ad hoc networks (MANETs) are wireless networks without infrastructure. Rather than connecting to a base station as in wireless telephony or an access point as in wireless local area networks, user radios collaborate to create their own network. When a user turns on his radio, that radio follows a protocol to announce its own existence and to discover other radios. The collective objective of the radios is to discover a topology that allows them to route communications amongst themselves. This approach to networking is necessary for tactical networks that cannot rely on infrastructure such as those used by military and disaster relief organizations. It is also appropriate for sensor networks and inter-vehicular networks.

MANETs present many challenges to networking protocols. In the absence of the central control performed by base stations and access points, protocols must provide distributed solutions. The mobility of nodes causes those solutions to be fleeting and so inter-radio communications for network maintenance persists throughout the lifetime of the network. This is no where more important than in the routing protocols that track network topology. The plethora of routing protocols proposed for MANETs attests to the challenge of this task. There is a direct correlation between the effectiveness of a protocol to track topology and the quantity of administrative traffic and this quantity increases as either the size or volatility of the network increases. However, wireless networks are capacity constrained. As a result, the bulk of the research in routing is directed towards reducing the overhead burden of disseminating topology data.

A major contributor to the quantity of overhead that is required to track topology can be attributed to the use of links to define topology. All MANET routing protocols that we are aware articulate the connectivity between pairs of nodes as links (Murthy and Manoj 2004). These protocols use the observation of past connectivity to infer connectivity in the future. In a typical proactive protocol, nodes periodically announce their existence and those that hear these signals assume a directed link between them. Next, nodes announce information on which nodes they can hear which supports determining bidirectional connectivity. Finally, nodes disseminate a comprehensive list of links, perhaps with a quality metric. Each node in the network takes the collective set of the observations of links and calculates its routing table. If a node discovers a link failure, it needs to resend a new list of links and all nodes that receive this information must purge the old list and replace it with the new followed by a recalculation of their routing table. The movement of a single node can cause it and multiple neighboring nodes to change their lists of links and then these lists must be disseminated throughout the network. In a typical reactive protocol, nodes needing to send packets to a destination send out a broadcast probe to discover the route. When the destination receives one of these probes it sends a reply back to the source. This reply either captures the route taken or specifies the next of the series of nodes that know the route to the destination. Here, not only are the links assumed because of past observations but also the whole path is assumed because it was observed in the past. If one link in the path fails because of movement or other reasons, a new path must be discovered, and the whole discovery process may need to be repeated. Since discovery often requires network-wide broadcasts it can be quite costly.

1.2 Node State Routing

Node State Routing (NSR) is the alternative to the link driven protocols described above. It uses a paradigm that more closely matches the physics of wireless networking. Rather than disseminating link states, nodes disseminate node states. A node's state is a set of observations made by a node of itself and its environment. Two pieces of information in the node state support the determination of connectivity in the network, the location of the node and a propagation map. The propagation map articulates how well a node can hear by direction and ultimately defines a space from which it can hear other nodes. Routers use a collection of node states to determine connectivity. Links are inferred between two nodes when they are within each other's listening space.

Similar to link driven protocols, nodes periodically send hello messages. These messages include the power at which the node transmits and the node's location. Distant nodes that hear the transmission record the location of the distant node, its own location, the time of the observation, and the observed path loss. Then, periodically or when the existing propagation map becomes incorrect, it generates a new propagation map from the collection of these measurements. The advantages of using NSR are that node states and propagation maps are concise when compared to lists of links, are unlikely to change for neighbors when a node moves thus reducing update volume, and they provide substantially more information than link states which can be used to support quality of service, energy conservation, and network management, and to anticipate not just track connectivity.

1.3 Propagation Maps

Nodes within an MANET communicate there connectivity through sending propagation maps inside their node state packets. These maps are transmitted in vector form referred to as a propagation vector. The structure of the propagation vector involves 8 bit words, which specify 256 different path loss exponents n ranging from 1.9 to 7.0 in increments that provide equidistant changes in propagation range. The propagation vector structure also divides a node's encompassing sphere representing connectivity range into 256 longitudes θ and 180 latitudes ϕ.

To understand the notation, consider each sector of the map to represent a specific path loss exponent n. If all sectors were explicitly defined by the propagation map, it would have the form (0, 0, n_{00}, θ_{01}, n_{01}, …, θ_{0x}, 255, ϕ_1, θ_{11}, n_{11}, …, θ_{1x}, 255, 180). Each sector covers the ϕ values ranging from the adjacent and previous ϕ from its n to the adjacent and following ϕ. Likewise, each sector covers the θ values ranging from the adjacent and previous θ from its n to the adjacent and following θ within the vector. However, if the previous θ is 255 then the range of θ values begins at 0. Also, the ranges of the θ and ϕ values for the first sector begin at 0.

Visually, the smaller n is for a sector within the map, the more connectivity there is, and the more outward the map extends those directions. To illustrate, an example of a propagation map is provided below.

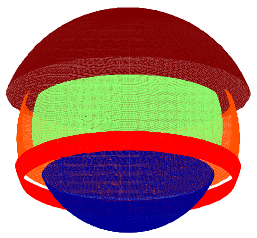

Figure 1: Illustration of Sample Propagation Map

Since $\phi = 0$, $\theta = 0$, $\theta = 255$, and $\phi = 180$ occur predictably, most of these elements can be dropped from the vector to obtain a more efficient structure. This is best described through an example. Consider the corresponding propagation vector to the above map in Figure 1, which is [2 255 69 4.5 100 3.1 255 100 2.7 255 110 4.7 0]. In this case, a path loss exponent n = 2 is assigned in directions containing a $\theta \in [0, 255]$ and $\phi \in [0, 69]$. A path loss exponent n = 4.5 is assigned in directions containing a $\theta \in [0, 100]$ and a $\phi \in [70, 100]$. A path loss exponent n = 3.1 is assigned in directions containing a $\theta \in [100, 255]$ and a $\phi \in [70, 100]$. A path loss exponent n = 2.7 is assigned in directions containing a $\theta \in [0, 255]$ and a $\phi \in [101, 110]$. Finally, a path loss exponent n of 4.7 is assigned in directions from containing a $\theta \in [0, 255]$ and a $\phi \in [111, 180]$. This notation is an efficient way for a node to convey its own information concerning connectivity. Of course, the fewer of bits required to define connectivity, the less network capacity required for sharing propagation maps among nodes in NSR.

That is the motivation for developing and studying the optimization algorithms to minimize vector length.

Using propagation maps has three constituent problems: collecting and pruning propagation observations in a manner that tracks the environment; building and then optimizing the size of the propagation maps; and disseminating the propagation maps in a manner that balances overhead with enabling a reasonable effective and reliable routing solution.

Five algorithms have been created to optimize the size of the propagation map. They involve a combination of brute force heuristics, linear programming, and genetic algorithms. A primary issue to determine is which algorithm provides the most optimal maps in a reasonable amount of time.

The simulation tool SHOPMET has been designed to investigate the best algorithm to deploy for NSR as well as the methodology for collecting propagation observations that best suits NSR. The rest of this paper is devoted to discussing the development and usage of the simulation tool SHOPMET.

2 SIMULATION ENVIRONMENT

SHOPMET simulates the generation of propagation maps for a single node in the network, which we will refer to as the center node in this paper. However, this still requires simulation of an entire MANET scenario. Simulating propagation map generation within a MANET for our study involves executing intensive algorithms requiring a Linear Programming solver and a genetic algorithm solver. These requirements make it difficult for a single software package to execute the entire simulation. Furthermore, we wish to display 3-D surface plots of the propagation maps, such as what is displayed in Figure 1 in the simulation outputs for visual demonstration and validation. SHOPMET contains three components to achieve these simulation goals. Two components are written in MATLAB and one component is written in C for use in OPNET.

2.1 Component Simulations

To begin we present an outline of the complete simulation environment. The first component, called Propmap Stimulus is responsible for node movement, the conduction of path loss measurements, and the sending of node state packets between nodes. Node movement is modeled through Brownian motion. There are parameter settings to define the frequency of sending node state packets and as well as collecting measurements from other nodes.

Propmap Stimulus has many other parameter settings to give the user flexibility in the network scenario he/she wants to replicate. The user can select an urban or rural environment. The urban environment consists of buildings, which effect path loss where as the rural environments contains fewer obstructions that increase path loss. If selecting an urban environment, the user can then select how many buildings and the dimensions and locations of each building in a scenario. The user can select the velocity of the nodes as well as specific parameters of the optimization algorithm among other parameter settings.

Inferring connectivity between two nodes in the simulation consists of determining if the estimated path loss is below a threshold for connectivity. It is not crucial to predict path loss with precision to infer connectivity (Stine and Veciana 2004). An estimate, such as, a log-distance path loss model is suitable. In SHOPMET, the model PL = PL(1 m) + 10*n* log(*d*) is used (Rappaport 1996) where PL(1m) is the path loss of the first meter, *n* is the path loss exponent, and *d* is the distance between the two nodes. Then an additional path loss is added for each meter that the signal traverses through an obstruction based upon the predefined coefficient of path loss for that object. This parameter can also be set by the user in SHOPMET.

In NSR nodes disseminate information about their propagation maps, locations, and other states in node state packets. SHOPMET simulates the distribution of node state packets. The transfer of these packets between nodes is contingent upon the existence of connectivity between two nodes. Likewise, path loss measurements from other nodes are contingent upon connectivity in SHOPMET.

When generating its propagation map, the center node also considers the location other nodes in the network that it doesn't have direct connectivity with through information contained in node state packets. These nodes are referred to as occlusions. The threshold for connectivity is set to 100 db in SHOPMET. When the center node learns about a new node for which it has no path loss measurement for, it assigns the path loss exponent in that node's direction to be .1 + the path loss exponent which would result in the threshold path loss of 100 db between the two nodes.

A screen shot of Propmap Stimulus is provided in Figure 2. Note that the unit distance on the screen is 10 meters. The yellow blocks represent the two buildings, which are present in this scenario.

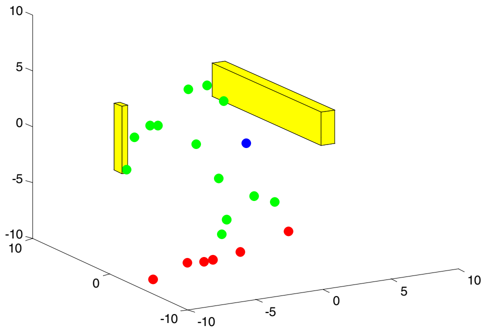

Figure 2: Screen Shot of Propmap Stimulus

Propmap Stimulus is then responsible for sending measurements repeatedly throughout the simulation to the simulation component, which is called NSR Sim, to manage the path loss measurements in propagation map studies. NSR Sim also will be the simulation tool for the study of NSR in complete network scenarios. NSR Sim is created in DLL libraries because of the other more technical simulation details that are required for the study of NSR, which will be performed in OPNET.

Propmap Stimulus receives formatted data concerning propagation measurements back from NSR Sim. It then calls the third simulation component, called Map Minimizer to create the propagation map using one of the five algorithms under investigation. It required the use of thee optimization toolbox and the Genetic Algorithm toolbox in Matlab. Map minimizer outputs the optimized map back to Propmap Stimulus to be plotted in 3D. The simulation then continues with more node movement, measurements, and transfer of node state packets until the next event for a propagation map creation. Figure 3 provides a chart to summarize all 3 component simulations.

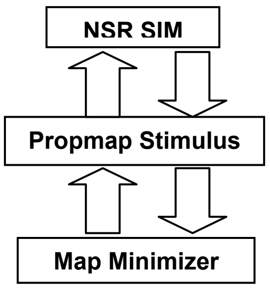

Figure 3: Chart of SHOPMET Simulation Components

2.2 Algorithms Tested in SHOPMET

The objective of each of the algorithms is to define a propagation map around the center node, which accurately infers connectivity to each node it is aware of in the network in the fewest number of bits possible. In other words, the algorithms attempt to solve for a propagation vector of minimum length. Lower and upper bounds are assigned in the optimization problem across small windows in space, which individually cover a single occlusion or observation. For observations, the minimum exponent is calculated using the observed path loss and the maximum exponent is calculated using the maximum path loss for a link. For occlusions the minimum exponent is calculated using a path loss that excludes the node from connectivity and the Maximum exponent is the maximum exponent value. An additional constraint that is considered is the goodness of fit between the path loss measurements and the propagation map's corresponding path loss for the sample points.

A brief outline of each of the give algorithms tested in SHOPMET is provided below. Nonlinear Programming is notated as NLP, Integer Programming as IP, Linear Programming as LP, and Genetic Algorithm as GA. This notation is provided below to specify the underlying optimization / search methodology.

1. **NLP**: Minimize length of vector such that goodness of fit is also considered as well as connectivity.

2. **IP**: Minimize length of vector such that connectivity is properly predicted at all sample measurement points.

3. **Two-phase cut (GA/LP)**: At each iteration cut a subset of the vector (containing no more than one n), solve for the remaining n, θ, ϕ values in the vector that minimize goodness of fit. Before each cut is made an LP is solved to smooth the n values in the vector and the subset containing the n with the smallest deviation to adjacent n is cut. Proceed with reducing vector until connectivity constraint cannot be achieved.

4. **Fast map (GA)**: Start with a vector of form [n, 0]. Using GA, solve for the best vector at the current length. If connectivity constraint cannot be achieved then increase the length of the vector. Continue until acceptable goodness of fit can be achieved.

5. **Constraint Based** – Simply iterate through all possible n, θ, ϕ values at each length, starting at length 3, until a combination is found that satisfies constraints on connectivity and goodness of fit.

3 INTERFACING THE COMPONENT SIMS

Algorithms for propagation map computation and routing were being developed independently of the work being done in SHOPMET for simulating and testing propagation map optimization and aging. These C-coded algorithms (henceforth known as the "NSR code") were built and tested within NSR Sim, an OPNET simulation environment. Although the NSR code was largely built to be portable (independent of underlying OPNET toolkits and operating system), the presence of profiling and memory allocation monitoring tools within OPNET was extremely valuable to effectively test, debug, and optimize the code. This meant that the resulting code was largely portable, but had some embedded OPNET dependencies to enable use of these execution and memory profiling tools.

One of the goals of the research effort was to integrate the underlying propagation map/routing code within multiple simulation environments (one of these being SHOPMET) being used to model and test various aspects of the MANET environment. Each of these environments was based on a proprietary closed-source toolkit that supported use of external dynamically-loaded libraries ("dll's" in the Windows environment) for extension and integration with third party code. In order to best leverage the work being done under OPNET, we decided to adopt the approach of embedding the NSR code within a dll that could be used with SHOPMET running under Matlab, using its built-in support for dynamically loading dll's.

Although this provided the basics of the integration mechanism, there was still an issue that the NSR code relied on a fairly low-level interface, utilizing pointers to C "structs" to pass data in and out of its functions. Moreover, some of these structs represent linked list structures. Although Matlab supports allocating and passing structs and arrays (with a few limitations), it is cumbersome to specify, create and manage data structures such as those required by the NSR code from within Matlab code. Another environment being used in a similar fashion (Runtime Revolution) made such usage even more complex, because it requires passing all data to external functions as strings. This led us to decide to develop an adapter layer that provided higher-level data types and a simplified API (application program interface) that could be exposed more effectively to third party tools such as Matlab. Basically, this interface involved specifying nodes by name and location, and providing calls to allow these nodes to be manipulated, to specify propagation loss observations between nodes, and to generate propagation maps, all based on symbolic node names.

We resolved the issue of OPNET dependencies by developing a simple replacement header file for OPNET functionality that would use a combination of C macros to eliminate profiling calls and replacement functions to map OPNET memory allocation to standard C library functions ("malloc"). We "coopted" the profiling calls to provide a runtime logging capability for debugging purposes (accomplished by further use of C macros). The dll itself was built using a combination of the Eclipse IDE with CDT (C development toolkit) and Mingw (Minimal GNU Windows toolkit) and gcc 3.4 to compile the C code. The benefits of this approach were that it helped to preserve portability of the code to platforms other than Windows for future implementations on embedded, real-time, and Linux-based systems.

The result of this integration effort was a single dll that could be built directly from the NSR code (unmodified from its form as embedded in an OPNET simulation) and that could be used by the SHOPMET simulation based on Matlab. A secondary dll that utilized the NSR dll was developed that provided a mapping of functions and string argument conversions for use with Runtime Revolution. This meant that all simulation efforts were based on the same underlying NSR code; that changes to the NSR code could be readily propagated to the other efforts simply be "dropping in" the revised dll; the provision of execution logging meant that debugging information could be captured in usage outside of the OPNET simulation.

4 EXPERIMENTATION WITH SHOPMET

To give the reader a better idea of how SHOPMET is used for the study of propagation maps and for NSR, we briefly discuss its input parameters and resulting performance measures. The input parameters for propagation map study involve the network scenario parameters for Propmap Stimulus discussed above as well as additional parameters pertaining to Map Minimizer. These include, for each algorithm, the time limit for the search for the optimal solution, the number of nodes contained in the network, and the minimal acceptable level of goodness of fit between the propagation map and the measurements at the sample points.

The performance measures of interest were the average length of the optimized map generated by the algorithms as well as the goodness of fit. Length is the most important criteria for choosing the best algorithm since there is a hard constraint for inferring connectivity within the formulation of the optimization problem. Experiments can be performed with this tool to investigate then, which algorithm performs best when the network contains a large set of nodes, is dense with buildings, fast moving nodes, etc. We can investigate which algorithms perform best under each the parameter settings discussed in the discussion of Propmap Stimulus.

5 CONCLUSION

The simulation SHOPMET is a flexible an effective tool for studying algorithms that generate propagation maps in MANET scenarios. It provides plotting of propagation maps and network animation to demonstrate the propagation map optimization algorithms in various network environments. In addition, it also contains the technical solvers to perform experimentation on the algorithm making it a powerful all in one tool for visualization and computation.

After the conclusion of this study on propagation maps, SHOPMET will be integrated within a future OPNET simulation to study large scale system designs of MANETs. In this application, SHOPMET will consist only of the Map Minimizer component. It will contain only a single algorithm for propagation map generation, which is the algorithm found in this study to be the most effective. It will be called iteratively during this MANET simulation to generate propagation maps for every node in the network. Each node will then use these propagation maps to compute an optimal route for a packet.

REFERENCES

Murthy, C. and B. Manoj. 2004. *Ad hoc wireless networks architectures and protocols*. Prentice Hall, Upper Saddle River, New Jersey.

Rappaport, T. 1996. *Wireless communications: principles and practice*. Prentice Hall, Upper Saddle River, New Jersey.

Stine, J and G. Veciana. 2004. A paradigm for quality-of-service in wireless ad hoc networks using synchronous signaling and node states. *IEEE Journal on Selected Areas in Communication*. 22 (7): 1301-1321.

AUTHOR BIOGRAPHIES

SCOTT L. ROSEN is a Senior Operations Research Analyst with the MITRE Corporation. He holds M.S. and Ph.D. degrees for Penn State in Industrial Engineering and Operations Research and a B.S. in Industrial and Systems Engineering from Lehigh University. His current work at MITRE involves simulation output analysis and optimization. He is a current member of INFORMS.

JOHN A. STINE received a BS in general engineering from the United States Military Academy at West Point, NY in 1981. He received MS degrees in manufacturing systems and electrical engineering from The University of Texas at Austin, Austin, TX in 1990 and later a Ph.D. in electrical engineering, also from The University of Texas at Austin in 2001. He served as an engineer officer in the United States Army for 20 years with relevant assignments as an assistant professor in electrical engineering at West Point and as a lead analyst in the Army's Task Force XXI experiment which was the Army's first attempt to network a brigade size maneuver force. He has been with The MITRE Corporation in McLean, VA, since 2001 where he does research in ad hoc networking and consults on projects concerning ad hoc networking, spectrum management, and modeling and simulation of command, control, communications, computer, intelligence, surveillance, and reconnaissance (C4ISR) systems. Dr. Stine is a member of the IEEE Communications Society and a registered professional engineer in the state of Virginia.

WILLIAM J. WEILAND is a Lead Software Systems Engineer with the MITRE Corporation. He holds an M.S. in Computer Science from the University of Maryland. His current research interests include simulation, visualization, and distributed computing. He has previously conducted work in human computer interaction and virtual environments.