International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
www.ijset.com, Volume No.1, Issue No.3, pg : 95-101 01 July 2012

Performance Analysis of Mobile Ad-Hoc Networks

Mr. Hradayesh Kumar Patel, Mr. Rajesh Shrivastava, Mr. Pradeep Kumar Dubey
SRIT, Jabalpur

Abstract: *We propose a novel, robust MANET protocols evaluation framework which enables researchers to track performance metrics and evaluate theoretical predictions. This framework speeds up the research and development spirals, provides faster feedback to algorithm developers and closes the loop between theory and qualitative analysis of the protocols' performance. Our test and evaluation effort is divided into two parts. Rapid prototyping and evaluation of proposed algorithms is performed in the MATLAB environment. These tools enable us to numerically analyze performance, capabilities, convergence, and robustness of new algorithms. The second higher fidelity approach is the test and evaluation framework developed in OPNET simulation environment. Its unique features are the novel application and evaluation process including sophisticated statistics collection and an event logging architecture.*

INTRODUCTION

Currently many ad hoc network routing algorithms such as AODV, DSR, are proposed, and plenty of researches have been conducted on their performance analysis [2][3]. However, most of these researches only deliver simulation results and their qualitative explanation, little has been done with regard to theoretical analysis. The reasons for this situation are partly due to the complexity and continuously change of network topology and many parameters that are highly dependent on particular scenario and difficult to abstract.

We propose MANET protocols evaluation framework as part of our Mobility-Aware Resource Coordination for Optimization of Network Infrastructure (MARCONI) effort to research, develop and evaluate a revolutionary Mobile Ad Hoc Network (MANET) prototype. The project requires radical rethinking of a wireless networking stack and has already led to prototyping and evaluation of new protocols. This collective effort spans a distributed team of researchers working together to translate groundbreaking theoretical research into significant performance gain over existing state of the art MANET. In order to track our performance metrics and evaluate theoretical predictions, we have created an evaluation framework that speeds up the research and development spirals, provides faster feedback to algorithm developers and closes the loop between theory and qualitative analysis of the protocols' performance. Our test and evaluation effort is divided into two parts. Rapid prototyping and evaluation of proposed algorithms is performed in the MATLAB environment. Our tools developed in MATLAB enable us to numerically analyze performance, capabilities, convergence, and robustness of new algorithms. The second higher fidelity approach is the test and evaluation framework developed in OPNET simulation environment. Its unique features are the novel application and evaluation processes we developed. These tools are independent of the type of networking stack being tested and thus allow for a direct comparison of various protocol iterations. The application module harness uses a scenario document easily created and imported into the simulation to allow for a flexible way of describing application scenarios from the tactical user's perspective. The statistics collection and logging framework we developed speed up the debugging cycle and help in evaluating the performance and the behavior of the new protocols.

PREVIOUS APPROACH

Network protocol development is a complex process riddled with design and implementation challenges. Assuring protocol correctness in all cases requires the programmer to not only understand the complexities of different parts of a protocol, but also gain insight into the interaction of the protocol within the network stack [4].

Furthermore, a distributed protocol development effort, while already challenging in its design stage, can be even harder during implementation and debug stages. Traditional software development methods, while successful at bringing the networking community a number of popular protocols, are not uniformly efficient in all possible types of development projects [5].

Typically, once a new protocol has been developed it needs to be simulated in a network simulation tool to evaluate its correctness and efficiency. OPNET Modeler is the industry standard for network modeling and 1 of 9. MOBILE AD HOC NETWORKS (MANET) PROTOCOLS EVALUATION FRAMEWORK simulation. It is based on a series of hierarchical editors that directly parallel the structure of real networks [6].

The standard method for processing performance data in OPNET involves graphical depictions of numerous statistics collected during the running of a simulation. Although statistics can give us coarse information about the performance of a given set of protocols, they cannot say anything about why a given protocol performed as such or where the problems are.

One other complication is the distributed nature of attributes defining a simulation. A scenario is described by attributes scattered over process, node, scenario and global settings all acting together. The application process module used in most standard models for packet-level traffic simulation. It is inflexible and strongly coupled with other standard process modules that many users choose to replace.

Finally, OPNET's basic statistics gathering architecture is insufficient to evaluate and explain large scale behavior. Though it offers a powerful event driven debugger, as the number of simulated nodes rises, it becomes very hard to keep track of events. It is difficult to store and compare events from different runs, or to customize their format [7,8].

Although OPNET does allow one to generate a visual plot of various events, we found the OPNET capabilities insufficient. The alternative analytic tool available to programmers is a protocol behavior log implemented as a series of console or file printouts. While useful for quick implementation checks, this approach is not viable for solving more complex problems due to size limitation on most platforms. Additionally, console output is difficult to search, and cannot be reused. Our project started out like many other advanced network research projects. A network simulation tool (we chose
OPNET) was an essential piece in our design. We assembled other tools for future debugging and evaluation that are mentioned above. However, having found the traditional way inefficient for our goals we invested our efforts in designing a more developed and streamlined process for simulation, evaluation, and debugging our protocols and algorithms. We believe that our approach has lead to a shorter development and evaluation cycle with a smaller team.

RAPID PROTOTYPING

The MATLAB environment provides an interface to easily script a prototype algorithm. Using this tool we can quickly code the approximation of algorithms derived from the theory and evaluate how well they perform [9]. The main purpose of this effort is to provide a numerical basis of confidence by solving the underlying optimization problem which guides further protocol development.
When creating the theory that underlies a network stack design we start with the statement of NUM (Network Utility Maximization) optimization problem which encapsulates the network utility and constraints. Our objective is to maximize the user perceived utility subject to the constraints on resources. We can specify, through optimization decomposition (OD) [10], an optimization problem for each component throughout the network stack. These optimization problems define coupling and information sharing requirements between different elements of the network. The downfall is that most of these problems are either NP hard or need to be solved in a centralized manner: thus our need for decentralized approximate solutions. There are many ways (heuristics) to find a solution to these problems; our aim is to find the one that does it the best. We can easily test and modify existing and new algorithms until we find one that suits our needs. This ability is very useful and helps us find algorithms that perform quite well.

The framework we developed consists of a main loop that steps though events. Each event can consist of position change, flow arrival or departure, or change in QoS. For each of these events we run an inner loop, which is on a small time scale, to simulate the packet exchange across the network. Throughout this inner loop we assume the node positions and applications remain fixed. For every event there are several modules that get executed. The first one is responsible for the network scenario, spatial distribution of nodes and their mobility. It also describes the types of flows that enter the network and their destinations. These are scriptable parameters which can be adjusted. The rest of the modules are responsible for implementing four major components: source rate control, routing, power control, medium (channel) access and flow scheduling. The choice of schemes that implement the above processes determines the type of network stack we simulate. The flows are simulated not as discrete packet flows but as continuous streams. This approximation allows us to model the system quicker and test the concepts which are the foundations of the new algorithms. We also compare the performance of a new algorithm such as priority based random access. We can show numerically the potential gains we are likely to get by implementing such an algorithm in OPNET. Because of the low fidelity of MATLAB we will most likely see less improvement when implemented in OPNET but it is a reasonably good predictor.

We visualize our results as graphs of various parameters of the network as compared to the control set of components. We implemented a basic 802.11 scheme for each of the variable components in our framework to gauge the improvements derived from our theory. For every simulation, we can run our both sets of components for the same flow and node distribution and mobility scenario. [Fig 1]

Another important capability, not related to simulating the network stack, is the visualization of topology, routing, and mobility. Our MATLAB framework is able to interface with OPNET and present routing and link visualization in a user friendly manner. We are able to import this data from OPNET simulation and visualize connectivity and how the

OPNET routing protocol implementation behaves in the context of node mobility. This capability has been very useful in verifying our routing algorithms. [Fig 5]

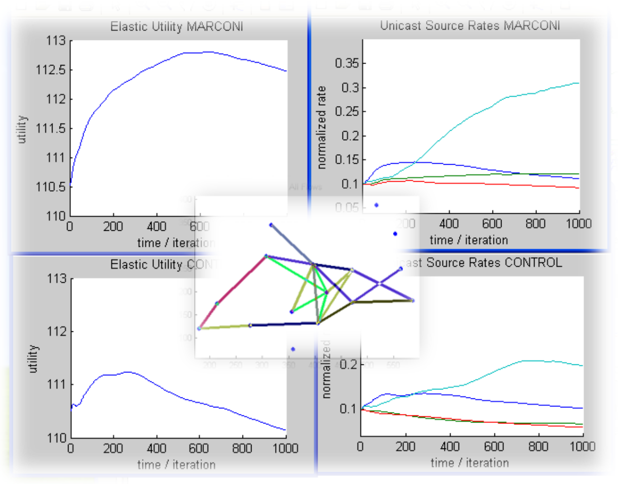


Fig. 1. MATLAB Framework Simulation Results.

Here, total network utility (left column) and unicast source rates (right column) network parameters are compared for a simulation using our set of components (top row) vs a control stack (bottom row). Clearly, we see improvement over the control stack. The center graph is the routing topology visualization at some time step. The blue dots are nodes and each colored path marks a route for a different flow.

OPNET SIMULATION ENVIRONMENT

The OPNET Modeler Wireless Suite provides high fidelity modeling, simulation, and analysis of a broad range of wireless networks. Technology developers leverage advanced simulation capabilities and rich protocol model suites to design and optimize proprietary wireless protocols, such as access control and scheduling algorithms. Simulations [11]

Key Features :

1. Fastest simulation engine among leading industry solutions
2. Hundreds of wired/wireless protocol and vendor device models with source code (complete OPNET Model Library)
3. Object-oriented modeling
4. Hierarchical modeling environment
5. Scalable wireless simulations incorporating terrain, mobility, and multiple pathloss models
6. Customizable wireless modeling
7. Discrete Event, Hybrid, and optional Analytical simulation
8. 32-bit and 64-bit fully parallel simulation kernel
9. Grid computing support for distributed simulation
10. Optional System-in-the-Loop to interface simulations with live systems
11. Realistic Application Modeling and Analysis
12. Open interface for integrating external object files, libraries, and other simulators
13. Integrated, GUI-based debugging and analysis [11]

A.Logging and event data collection

Our logging infrastructure provides a powerful way for any module in the simulation to report important events – whether this is route table changes, traffic flow beginning or something more fine grained. Unlike statistic collection it is able to collect arbitrarily complex event objects. Each event combines relevant information described by the developer: time stamp, initiator node id, packet id, or any other atomic piece of information relevant to the event.

From software development perspective, the logger class is a fully standalone module which can be defined by a test and evaluation team independent of the code to be analyzed. The responsibility of logging events is left up to the protocol developers who use the logger features via a single static function throughout their code. Once created and logged (via a static initialization function), the events can be outputted into a variety of formats: a command window output stream, a plain text file or an XML file format, or an excel spreadsheet in tabular form. There is a robust inheritance hierarchy in place which allows sub-classing of logging events. For example, a general routing event can have other children: route discovery initiation event, a node’s routing table update event, routing packet receipt event, etc. This allows complex filtering to allow the experimenter to focus on particular types of events. This can be done in one central location regardless of the complexity of the rest of the system. The power of such flexible output is apparent. Once the events are placed into spreadsheets, they can be further filtered and sorted by time, or by any other field. With the output of just one simulation the data can be analyzed chronologically then per wireless node, etc.

B.Application process with xml scripting of Scenarios

Our project relies on flexibility of simulating various traffic flow patterns to test many specific features of the new protocols from the perspective of the end user. For example, ‘a voice flow at 8kbps should be sent with high urgency and quality of service demands to a group of receivers’.

In OPNET native application module, this requires translating application behavior into traffic patterns, changing application parameters accordingly and saving these profiles for use by other nodes – a cumbersome process. Modifying these patterns might be a time consuming task even for a small behavior change. As a solution, we designed a more light-weight application suite as an OPNET module. An application dispatcher interfaces with the protocol lower in the stack, the transport protocol [Fig. 2], and starts child processes for applications when necessary. We currently support the following application types:

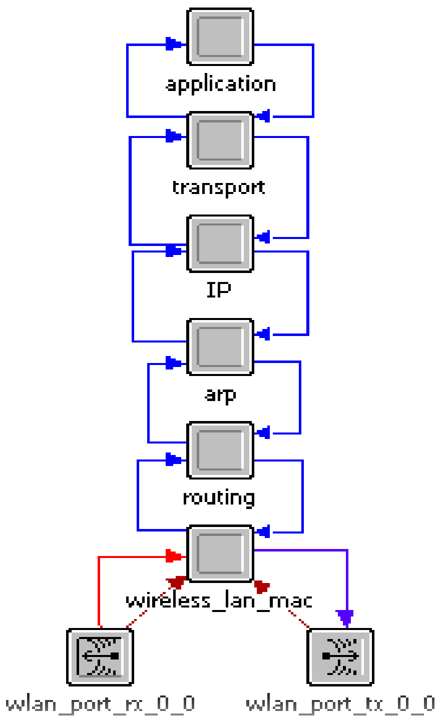


Fig. 2. MARCONI Application Harness.

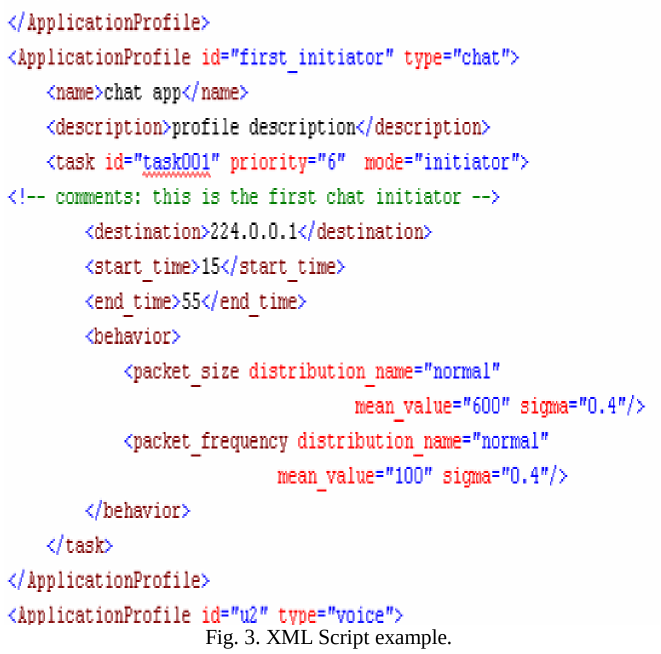
The pictured networking stack is modeled in OPNET and includes our own application process. The plug-an-play interface of the application harness enables integration with different stacks to be evaluated against each other's performance.

1. **File Transfer:** transferring files of specific size There are no constraints on the service, and throughput and delay are allowed to vary arbitrarily, as long as the file takes to be delivered. The initiator side chooses a file size to transmit and schedules itself to transmit packets periodically until done. The receiver simply records them.

2. **Chat:** sending text bursts The chat application transmits two-way low data rate bursty traffic. We script the initiator task to start at a particular time and the receiver starts responding with its own flow of data once it receives the first packets thus initiating a chat conversation.

3. **Voice:** sending VoIP This inelastic application implements a non-trivial rate constraint and specifies a tight delay constraint. Its operation and traffic patterns are similar to chat, though of higher bandwidth.

The script schema allows us to design application profiles to mimic the behavior of each application that a tactical user (i.e. warfighter, soldier) would be using minute by minute. In the future, we plan to also model video streaming applications, short command and situation awareness messages. Together with the above application types we already have implemented these are the applications typically used by a warfighter [12]. For each simulated user (node) in the network each application type loads a corresponding (by type) application profile. This can be the same application profile for all users or an individual one. Our application scenario description can thus be very general or very granular depending on the requirements of the test cases. In addition, this approach satisfies our desire for a single location where multiple applications could be easily scripted and their profiles saved for distribution to others [Fig. 3].



```
</ApplicationProfile>
<ApplicationProfile id="first_initiator" type="chat">
    <name>chat app</name>
    <description>profile description</description>
    <task id="task001" priority="6"  mode="initiator">
<!-- comments: this is the first chat initiator -->
        <destination>224.0.0.1</destination>
        <start_time>15</start_time>
        <end_time>55</end_time>
        <behavior>
            <packet_size distribution_name="normal"
                                    mean_value="600" sigma="0.4"/>
            <packet_frequency distribution_name="normal"
                                mean_value="100" sigma="0.4"/>
        </behavior>
    </task>
</ApplicationProfile>
<ApplicationProfile id="u2" type="voice">
```

Fig. 3. XML Script example.

Each Application profile describes a particular application's sequence of tasks and their behavior as it would be observed by the user. It can be scripted using a predefined schema to define the behavior of the simulation as well as provide a clear story to a human reader. Here the chat application is defined to start a flow at 15 secs and finish it at 55 secs while sending approximately (defined by the normal distribution) 100 packets per second each of size about 600 bits.

THE NEW PROCESS

The research effort is broken up into several Spirals. Each spiral includes the evelopment of a new piece of theory, its analysis and subsequent implementation and testing. At the end of each spiral we release a code base which we evaluate to isolate the performance improvements. The starting point of each Spiral is the draft of theoretical innovations that would be needed to improve the performance of the MANET from the perspective of a tactical user.

The performance improvement we consider would be the improvements in network throughput, reduced latency, and greater reliability as perceived by the end user – the war fighter.

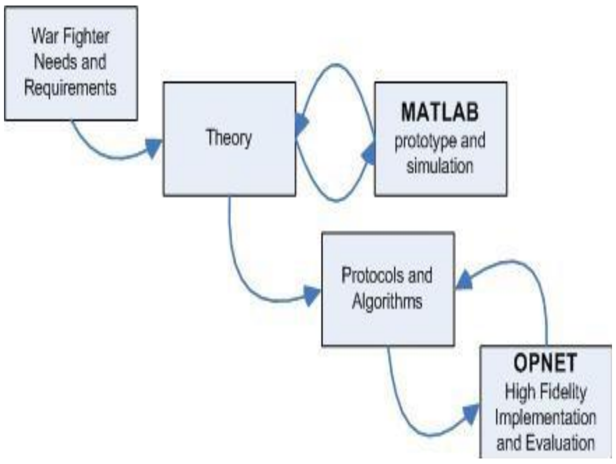

War Fighter Needs and Requirements
Theory
MATLAB prototype and simulation
Protocols and Algorithms
OPNET High Fidelity Implementation and Evaluation

Fig. 4. The 'waterfall' process which describes each spiral of our research project. End user requirements drive the theory behind our protocols which is then quickly evaluated using the MATLAB tools we developed. The Protocols and new features result. A higher fidelity simulation completes the process.

Another example is our MATLAB tools built specifically for visualization of Routing topology established as a result of the simulation run. We are able to replay the simulation as a movie watching for the available connectivity and route establishment as a result of this connectivity [Fig 5].

Once the bugs are fixed we are ready for the final stage of the spiral – evaluations. Again we employ our application module to design more complicated and more realistic scenarios. We design the node mobility and each node's application behavior as it would be seen by each user. Each of the applications define their own behavior as specific as "send a voice message at 10 seconds to multicast IP 224.0.0.1" or "reply
to all incoming voice traffic for IP 224.0.0.2".

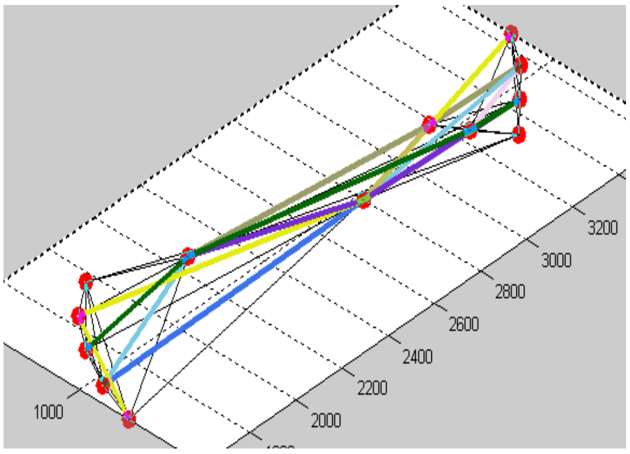


Fig. 5. MATLAB Routing Topology Visualization.
Here the links between nodes are shown as thin grey lines. The thicker color coded lines are routes at a particular time instant. The tool can be played out as an animation or stepped through chronologically.

We then apply these application attributes to both the new network stack we are evaluating and the baseline network stack we chose at the beginning of the project as our starting point. We thus run two simulations for each setup. We then gather local (per node), and global (per network) statistics outlined above to see what performance gains we notice as a result of our innovations. Furthermore, specific features can be turned on and off to pinpoint the improvement results and tie them to specific innovations. At our discretion, we also run a third simulation for the same setup on the networking stack resulting from the previous spiral. This serves as regression testing and aids our analysis of features contributing to the performance gains.

CASE STUDY

In this section we will take a representative siege scenario and evaluate it in MATLAB and then in OPNET. As part of the MARCONI program objective we are required to provide "equivalent performance at 10% of the bandwidth" so our evaluation method has to reflect the effect of bandwidth on performance. For each scenario we evaluate the bandwidth is reduced until the stack can no longer support the load, we call this the saturation bandwidth. We apply this method to both the MARCONI stack and a representative Control stack; the ratio is the percent of bandwidth which we achieved equivalent performance.

When we evaluate the performance of the stacks on the above scenario we have two metrics we are concerned with. The elastic utility is the sum of the logs of each of the elastic flows (in bps). For inelastic flows the utility is the sum of the valid flow periods (or brownie points); a flow receives a brownie point if it reaches it destination at or above 90% of its min rate.

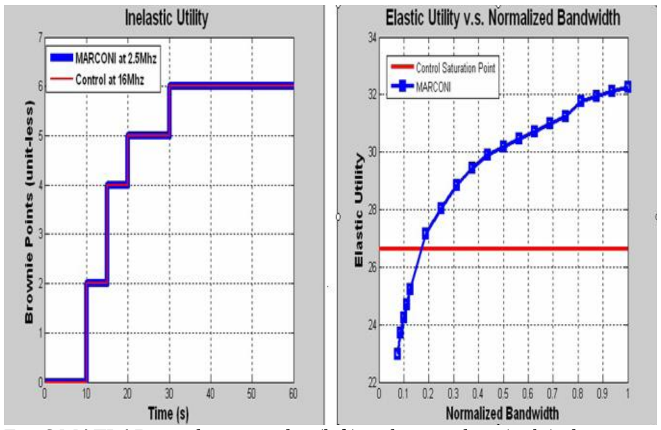


Fig. 6 MATLAB simulation results, (left) inelastic utility, (right) elastic utility.

The MATLAB results are shown in [Fig 6]. The inelastic utility (left) for the MARCONI and Control Stacks are plotted for each of the respective saturation bandwidths. The elastic utility (right) shows MARCONI's elastic utility plotted against the fraction of the Control stacks bandwidth. The threshold line (red) indicates the utility the Control stack achieved at saturation. What these two plots tell us is that MARCONI was able to carry the offered load at 2.5 Mhz where the Control needed 16 Mhz. This equates to equivalent performance at about 16% of the bandwidth. These are great results but have a few caveats due to the implicit low fidelity of the MATLAB simulation tool.

Our next step, once we have verified that the proposed algorithms perform well, is to determine the changes that need to be made for a real world implementation. The performance in MATLAB gives us somewhat of a best case of how good the proposed algorithms can perform. Once we move to OPNET modifications and approximations must be made in order to implement them as protocols.

The OPNET results shown in [Fig 7] are very similar to those shown in MATLAB. The MARONI and Control stack were saturated at 4.8 Mhz and 19 Mhz, respectively. This equates to equivalent performance at about 24% of the bandwidth. The discrepancies between the elastic utility plots is due to the fact that in OPNET we used a log base 10 and in MATLAB we used a natural log.

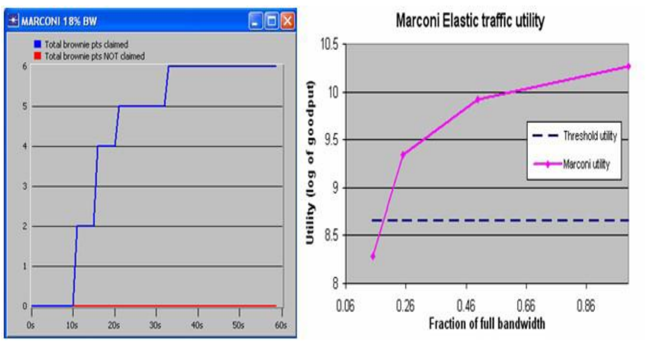


Fig. 7 OPNET simulation results, (left) inelastic utility, (right) elastic utility.

This case study clearly shows how we use MATLAB to find and evaluate potential protocol algorithms. These algorithms are molded in MATLAB till they have the desired properties to present a feasible real world implementation. This implementation can then be coded in OPNET and eventually move to a real radio.

FUTURE WORK

We plan to add a few more capabilities in the near future. To begin with, we intend to implement a few other application types: video streaming, short messaging, and situation awareness messages. One other idea we have been nurturing is to implement a central run-time application profile distributor to allow batch mode execution of multiple simulations with different (e.g.randomized) traffic profiles. We hope to create a central modeling process that can allocate application profiles to nodes at runtime based on a single configuration. This process will read a master script that describes probabilistic distributions specifying which nodes may run which application profiles and with which parameters. This will greatly enhance our ability to run sensitivity and confidence tests.

CONCLUSION :

We have presented our novel, robust MANET protocols evaluation framework which has enabled us to dramatically speed up the research and development cycle of our effort, improve the efficiency of the theory to protocol cycle iteration, and otherwise increase the productivity of our research team spanning over 6 public and private research institutions. Our rapid prototyping framework in MATLAB has enabled us to numerically analyze performance, capabilities, convergence, and robustness of a new network stack before a more thorough implementation effort is required. Our higher fidelity simulation and evaluation framework has enabled us to test the network stack programmatically and with higher accuracy. We believe it has enables us to discover the design and implementation flaws much faster than otherwise would be possible. This has contributed to the overall efficiency of our research effort.

REFERENCES:

1. Vadim A. Slavin, Mike Wittie, Michael Polyakov MOBILE AD HOC NETWORKS (MANET) PROTOCOLS EVALUATION FRAMEWORK

2. C. E. Perkins editor, “Ad hoc networking”, Addison - Wesley, 2000

3. C. E. Perkins etc, “Performance comparison of two on-demand routing protocols for ad hoc networks”,

4. D. Lapsley, M. Bergamo, "An integrated approach to the development of wireless network protocols", in Proc. of the 1st international Workshop on Wireless Network Testbeds, Experimental Evaluation Characterization ,Los Angeles, CA, USA, 2006). WiNTECH '06. ACM Press, New York, NY, 10-17

5. David Cavin, Yoav Sasson, Andre Schiper, "On the accuracy of MANET simulators", in Proc. of the second ACM international workshop on Principles of mobile computing, New York, United States 2002, pp. 38–43

6. Modeler Wireless Suite for Defense [web page]. 2006 OPNET Technologies,
Inc. Available: http://www.opnet.com/solutions/network _rd/modeler_wireless_defense.html

7. X. Chang, "Network simulations with OPNET", in Proc. of the 31st Conference on Winter Simulation: Simulation---A Bridge To the Future - Volume 1, Phoenix, Arizona, United States, 1999.

8. Varshney, M., Xu, D., Srivastava, M., and Bagrodia, R., "SenQ: a scalable simulation and emulation environment for sensor networks", in Proc. of the 6th international Conference on information Processing in Sensor Networks, Cambridge, Massachusetts, USA, 2007.

9. Elphick, D., Leuschel, M., and Cox, S., "*Partial evaluation of MATLAB*", in *Proc. of the 2nd international Conference on Generative Programming and Component Engineering*, Erfurt, Germany, 2003.

10. L. Bui, R. Srikant, A. Stolyar, "Optimal Resource Allocation for Multicast Flows in Multihop Wireless Network," npublished.

11. http://www.opnet.com/solutions/network_rd/modeler_wireless.html

12. Joe Leland**,** Future Army Bandwidth and Capabilities, Rand Corporation, 2004