An Investigation Into Improving Runtime Performance in Evaluating a Mobile Ad-Hoc Operational Scenario Using OPNET

G. Comparetto, M. Mirhakkak, D. Houser, B. Hung, N. Schult , R. Wade, All of The MITRE Corporation

Abstract--- Quantifying the end-to-end performance of evolving DOD communication networks is highly desired by the component and network designers during all phases of the development process. Analytical techniques, in-lab testing and field demonstrations are all necessary toward this end but all have limitations in addressing this need. Simulation remains a primary method with which to generate end-to-end performance. However, simulation often results in unacceptably long runtimes for these types of networks. The purpose of this paper is to report on the results generated in the second year of a 2-year IR&D program to investigate methods to improve simulation runtime performance when simulating mobile ad-hoc communication networks.

I. INTRODUCTION

The complexity of evolving DOD communication networks continues to grow. At the same time, the need to support the full range of user platforms configured as ad-hoc communication networks remains as a pivotal requirement towards the development of a seamless network-centric communications infrastructure. Quantifying the end-to-end performance for these networks continues to be highly desired by the component and network designers during all phases of the development process. The complexity of such networks rarely allows one to generate this performance analytically through the application of closed-form expressions. Additionally, in-lab testing and field demonstrations have limitations in terms of scalability and cost, leaving simulation as a primary method with which to generate end-to-end performance.

The simulation of ad-hoc communication networks can be challenging from a runtime performance standpoint. A variety of network characteristics contribute to this including mobility, offered traffic load, the number of nodes, the traffic mix (i.e. voice, data, video), support for multicast traffic, and the need to account for terrain. We have been investigating methods of improving runtime performance as part of a 2-year internal research and development effort. Our initial research in this area began in September 2004 and was reported at OPNETWORK 2005 and MILCOM 2005 [5]. The purpose of this paper is to complete our report on the results generated from investigating a variety of mechanisms to improve simulation runtime performance of simulations of mobile ad-hoc communication networks.

II. METHODOLOGY

In FY05, we accomplished the following:

1. Developed a baseline 114-node operational scenario including 104 fixed and mobile ground nodes and 10 Autonomous Air Vehicles (AAVs). Two levels of offered traffic loads were investigated - 39 kbps and 450 kbps. Each included a mix of data, voice, and video. Over 90% of the offered traffic was comprised of multicast traffic.
2. Developed a baseline node model that used IEEE 802.11 at the MAC layer, Optimized Link State Routing Protocol (OLSR) [3] to support unicast routing, and Protocol Independent Multicast – Sparse Mode (PIM-SM) [4] to support multicast routing.
3. Generated runtime performance results using our M&S Environment and End-to-End M&S Testbed (EMAST) [1-2].
4. Generated and evaluated detailed profiling data for the Baseline Scenario.
5. Quantified the degree of runtime performance improvement achievable from S/W compiler selection and the application of hybrid simulation techniques.

The reader is referred to [5] for more detailed descriptions of the baseline operational scenario, node model, profiling data and runtime results generated during the first year of this 2-year IR&D effort.

In FY06, we extended our research in the following areas:

1. Investigated the impact of existing features available through the OPNET simulation tool (e.g., packet copy, packet duplication,

wireless domain, and dynamic receiver groups).
2. Extended the profiling data based upon our observations in order to identify other “promising” areas on which to focus our efforts.
3. Based on #2 above, investigated the impact of (a) modifying the OLSR parameters set and/or the routing table formation algorithm(s), (b) increased mobility, (c) the introduction of the Path Loss Matrix (PLM) methodology, and (d) parallel processing.

III. IMPACT OF EXISTING FEATURES AVAILABLE IN OPNET

Packet Copy Early in our research, we were actively pursuing methods to decrease the memory footprint of mobile ad-hoc communication networks in addition to runtime. Packet copy is a technique in OPNET to reduce the memory footprint of a simulation by only reproducing the header of a packet upon implementing a copy command. We investigated the performance impact of various Packet Copy configurations. The improvements in memory footprint for all cases were negligible, with a slight increase in runtime for several configurations. The runtime results are shown in Figure 1. We discovered that the Packet Copy technique is best applied if the packets are broadcast, have a large number of named fields and/or if there are nested packets – none of which applied to our baseline scenario.

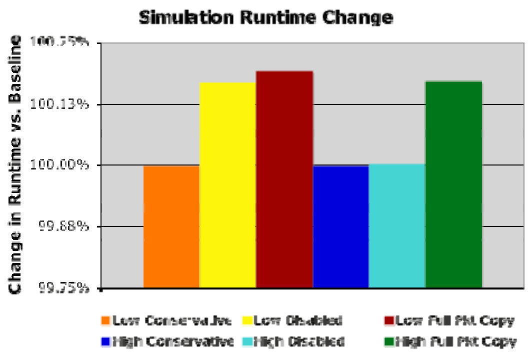

Figure 1: Runtime Results Using Various Configurations of Packet Copy

Packet Duplication Packet Duplication is a technique in OPNET used to copy packets only after successful completion of the Closure and Channel Match radio pipeline stages. Theoretically, this would avoid wasteful copy and deletion of packets that fail Channel Match. We investigated the performance impact of various Packet Duplication configurations. The improvements in memory footprint and simulation runtime for all cases were negligible. We discovered that the Packet Duplication technique is best applied if the scenario is comprised of multiple radio device types that cause numerous Channel Match failures – again, none of which applied to our baseline scenario.

Wireless Domain Wireless Domain is a technique in OPNET to cache physical layer computation results between a pair of geographically defined regions to reduce simulation runtime. We investigated OPNETs full-grid wireless domain feature in which the entire geographical space of the scenario is divided into an evenly divided square N x N grid. Our study included four experiments, N = 1, 10, 32, and 64 (6 seeds each experiment). While the total memory footprint of the simulation increased as the number of cells in the grid increases, there was minimal impact on overall accuracy in terms of completion rates. In fact, the variation in seeds had more impact on the results than did the actual grid size. In addition, the experiments revealed no significant impact on runtime or total number of events. Extensive profiling results, detailed in [5], revealed that a minimal amount (less than 2%) of the total runtime is spent in the radio pipeline stages for this scenario; hence, there is little room for runtime improvement by attempting to optimize the physical layer.

Dynamic Receiver Group Dynamic Receiver Group is a technique in OPNET that allows the user to define groups of receivers (using either physical or logical definitions) that are eliminated from traversing the radio pipeline thereby resulting in reduced runtime. Due to the combination of the insignificant improvement with the wireless domain experiments achieved, as noted above, and the profiling results of our scenario which indicated such a small amount of runtime being expended in the radio pipeline, we concluded that it was not necessary to study the impact of receiver groups with this scenario.

IV. EXTENDED PROFILING RESULTS

We set out to accomplish two goals using profiling in FY06: (1) Quantify the impact on the radio pipeline by using the Path Loss Matrix

(PLM) methodology[1] [5] versus dynamic inline TIREM processing, and (2) Utilize profiling to support our investigation into improving runtime by optimizing the OLSR route calculation algorithms.

Quantify Radio Pipeline Impact of PLM, FS, and LR In comparing inline Freespace (FS) path attenuation (i.e., calculating path attenuation within the OPNET simulation) with the offline path loss matrix (PLM) technique, we learned in FY05 [5] that FS resulted in a much longer runtime than PLM, but with a much smaller proportion of that runtime consumed by the pipeline. We reasoned that this was due to an increase in connectivity because of the lack of terrain interference, which required less pipeline attenuation processing, but substantially more routing updates.

Since profile runs took on the order of 3 to 6 days, we used only low offered traffic load when profiling. And building on the benchmark profiling conclusions of the previous year [5], we ran simulations to 1000 seconds rather than the full 4500 seconds, to further reduce the runtime of experiments. OPNET v11.0 was the primary version for these runs for the sake of consistency.

Table 1 summarizes the experiments that were profiled. To begin with, FS-Line of Site (FS-LOS) varied slightly as compared to FS (no occlusion). As expected, the results using OPNET 11.0 showed that the dynamic inline terrain methods (FS, TIREM3, LR) resulted in the pipeline consuming a higher proportion of runtime than the PLM. However, with OPNET 11.5 optimizations, TIREM 4 gave precisely the reverse. Examining these 11.5 results in greater detail, we identified an efficiency threshold parameter implemented by OPNET (although not documented) and assigned in the TIREM4 module. This parameter has the effect of dropping packets whose path loss exceeds the threshold.

Figure 2 provides additional detail regarding how pipeline stages vary in relative runtime consumption. As shown, the Closure (dra_closure) and Power (wlan_power) stages consume the greatest proportion of runtime.

Optimizing OLSR Route Calculation Algorithms

[1] The PLM is created offline a-priori using DTED terrain data and TIREM.

Our previous profiling results [5] revealed OLSR's proactive routing modules to be the top consumers of simulation runtime. This year we performed more detailed profiling to support our investigation into improving runtime by optimizing the OLSR route calculation algorithms.

Table 1: Profiling Results Using The Path Loss Matrix (PLM), Free Space (FS), TIREM, and Longley Rice (LR)

Path Attenuation Method	Opnet Version	Inline Terrain Data?	Run-Time in Pipeline			
			Run 1	Run 2	Run 3	Average
PLM	v11.0	--	1.1%	1.2%	--	**1.1%**
PLM	v11.5	--	2.2%	--	--	**2.2%**
FS-LOS	v11.0	--	1.1%	--	--	**1.1%**
FS-LOS	v11.0	YES	1.8%	1.8%	1.5%	**1.7%**
TIREM 3	v11.0	YES	1.9%	1.7%	--	**1.8%**
TIREM 4	v11.5	YES	0.5%	0.3%	--	**0.4%**
LR	v11.0	YES	3.0%	2.9%	2.9%	**2.9%**

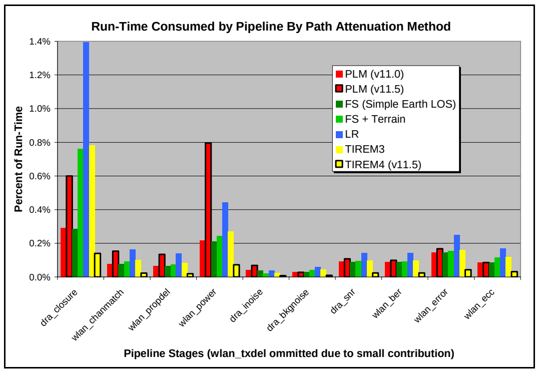

Figure 2: Relative Pipeline Stage Runtime For Various Path Attenuation Configurations

Sections of the OLSR route calculation routine were profiled independently to compare each optimization with the baseline and to previous optimizations. This effort helped us to identify the most time-consuming modules upon which to focus our optimization efforts.

Table 2 shows the top ~60% of runtime consumer modules resulting from profiling several configurations developed to improve the OLSR route calculation efficiency. These configurations are based on the Breadth-First Search (BFS) Algorithm and are referred to as Mod1, Mod2ONI and Mod3ONI in Table 2. Additional detail regarding the BFS algorithm and these configurations can be found in [6].

The chart shows that in Mod1, 9% of runtime occurs in the highlighted *olsr_rte_calculate* routine, whereas in Mod2ONI, that time is reduced to 6.3%[2]. This data confirms that the most expensive module in the simulation has been reduced to the third most expensive module after optimizations were done.

Table 2: Top ~60% of Consumers Identified by Profiling

Function Name	Function Time (module only)		
	Mod1	Mod2ONI	Mod3ONI
olsr_rte_calculate_route_table	9.0%	6.3%	6.6%
Inet_Cmn_Rte_Table_Entry_Add_Options	7.0%	7.7%	7.2%
olsr_rte_process_hello	6.4%	7.6%	7.2%
op_prg_list_access	4.4%	4.2%	4.3%
olsr_rte_neighborhood_topology_check	3.2%	2.6%	2.2%
olsr_rte_expired_two_hop_neighbor_entries_remove	2.7%	3.4%	3.1%
oms_ptree_entry_add	2.6%	2.9%	2.8%
olsr_rte_calculate_mpr_set	2.5%	2.9%	2.6%
inet_rtab_index_to_addr_convert	2.4%	1.4%	1.3%
olsr_rte_two_hop_addr_reachability_check	2.0%		2.1%
wlan_interrupts_process	2.0%	2.1%	2.1%
wlan_power_mod2_mt	1.9%	2.1%	2.1%
wlan_mac	1.6%	1.7%	1.7%
olsr_rte_calculate_reachability	1.5%	1.6%	1.4%
olsr_rte_calculate_degree	1.4%	1.5%	1.4%
oms_ptree_node_destroy	1.3%	1.5%	1.4%
ip_cmn_rte_table_entry_free_proc	1.3%	1.5%	1.3%
ip_rte_central_cpu	1.2%	1.3%	1.3%
ip_rte_central_cpu_packet_arrival	1.0%	1.1%	1.1%
ip_cmn_rte_table_entry_free	1.0%	1.1%	1.0%
ip_rte_packet_arrival	0.9%	0.9%	0.9%
wlan_physical_layer_data_arrival	0.7%	0.8%	0.8%
ip_rte_datagram_higher_layer_forward	0.6%	0.6%	0.7%
ip_pim_sm_mod1	0.5%	0.5%	0.5%
	59.2%	57.1%	57.5%

V. OLSR

Our detailed profiling information showed that around 50% of the processing time was spent performing route forwarding table calculations. Unlike wired networks that summarize addresses and produce a small number of entries in the forwarding table at each node, MANETs require an entry for each wireless interface and each node's table must be recalculated whenever there is a slight change in the network.

The above observation encouraged us to look for an efficient algorithm for calculating route forwarding tables. The details of this algorithm are reported in [6].

We conducted several experiments with different versions of an efficient route forwarding table calculation algorithm which is based on the Breadth-First Search (BFS) Algorithm. The results of this study are shown Figures 3 and 4. As Figure 3 shows, we were able to reduce the simulation run time by a maximum of 27%. Figure 4 shows the completion rate for unicast, multicast, and combined traffic for all the BFS configurations considered in this study.

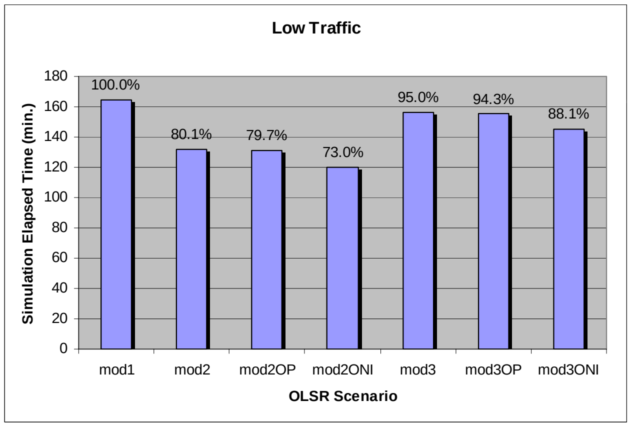

Figure 3: Simulation Runtime Results For Various Configurations of the BFS Algorithm

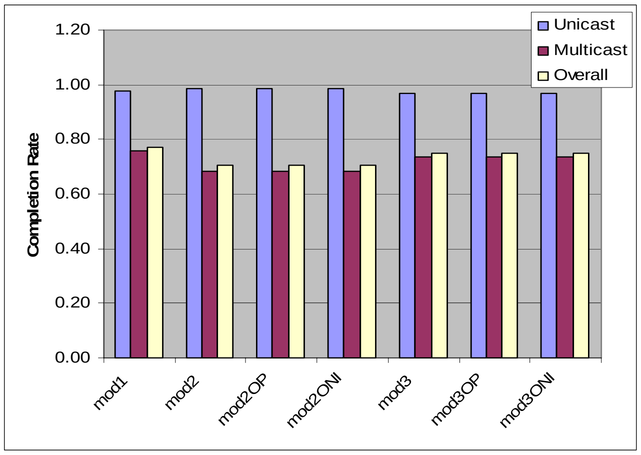

Figure 4: Completion Rate Results For Various Configurations of the BFS Algorithm

In addition to modifying the forwarding table calculation algorithm, we investigated the impact of several key OLSR protocol parameters had on both simulation runtime and accuracy. These parameters included Hello Interval, TC Interval, Neighbor Hold Time, Topology Hold Time and Duplicate Message Hold Time. The experiments run are shown in Table 3 while the resulting simulation runtime and completion rate results are shown in Figures 5 and 6, respectively.

[2] This does not map directly to the absolute simulation runtime improvement because these values do not include time spent in the modules' sub-function calls, or potential side-effects of the modifications that might be realized in other routines.

Table 3: Experiments Run To Investigate Impact Of OLSR Parameter Values On Runtime

	Hello Interval	TC Interval	Neighbor Hold Time	Topology Hold Time	Duplicate Message Hold Time
OLSR_Test_0	2	5	6	15	30
OLSR_Test_0point5	2.83	7.07	8.49	21.21	42.43
OLSR_Test_1	4	10	12	30	60
OLSR_Test_1M	4	10	12	30	30
OLSR_Test_1point5	5.66	14.14	16.97	42.43	84.85
OLSR_Test_2	8	20	24	60	120
OLSR_Test_2point5	11.31	28.28	33.94	84.85	169.71
OLSR_Test_3	16	40	48	120	240

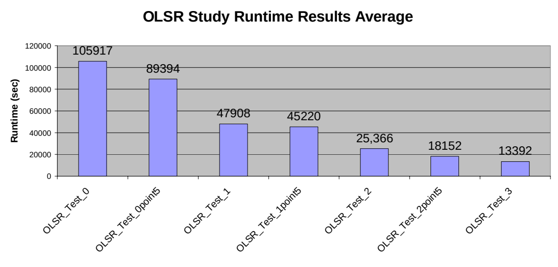

Figure 5: Simulation Runtime Results for OLSR Parameter Runs

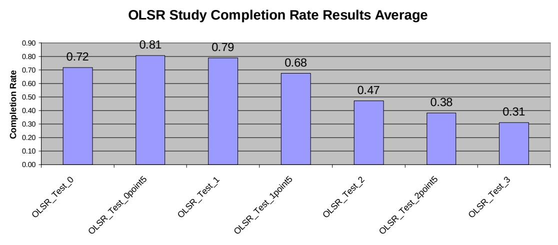

Figure 6: Completion Rate Results for OLSR Parameters Runs

As shown in Figure 5, modifying the OLSR parameters can significantly influence runtime. Our results indicated that the simulation runtime scales with the OLSR Hello Interval. However, the runtime improvement comes at the expense of accuracy as shown in Figure 6. It would appear that increasing the OLSR Hello Interval may only be an option to improve runtime in special situations such as trend analyses in which relative performance (versus absolute performance) may be adequate.

VI. IMPACT OF MOBILITY

Our goal here was to quantify the impact of mobility on simulation runtime. To do this, we modified our baseline scenario to significantly increase the amount of mobility in the scenario. We then made a number of runs and compared the simulation runtime attained with the baseline scenario. To our initial surprise, the additional mobility did not result in a significant increase in simulation runtime. Upon further reflection, this should have been expected. The predominant factor attributing to simulation runtime was found to be the OLSR route formation calculations based upon our profiling results. Furthermore, based upon the results in the previous section, the simulation runtime was directly proportional to the OLSR parameter Hello Interval which is a parameter that defines the periodicity of the OLSR route formation calculations. Since we used the same OLSR parameters set values for both the Baseline Scenario and the Highly Mobile Scenario, it is logical that the simulation runtime would not change significantly which is precisely what we observed.

VII. IMPACT OF PLM METHODOLOGY

Originally, our path loss matrix was a part of the power computations at the receiver (i.e., the power stage of the radio pipeline). We studied the impact of moving the PLM to the transmitter closure computations. When moving the PLM to closure, a threshold was also defined to be consistent with OPNETs TIREM implementation which was identified earlier. This threshold was designed to eliminate receiver computations (SNR, BER, etc) for links where the path loss was large enough to prevent the signal from being coherently received by the receiving terminal. The study revealed a 6% increase in runtime performance with little impact on completion rates, when the path loss information was used in the transmit portion of the pipeline stages.

We also quantified the impact of applying the PLM methodology with the use of TIREM3 and TIREM4 inline during the OPNET simulation run. An experiment was set up in which we ran the Baseline Scenario using the PLM methodology or TIREM3 using OPNET 11.0. These runs were then repeated using OPNET 11.5, with additional runs using TIREM4 (not available until OPNET 11.5). The results generated indicated that using TIREM (either TIREM3 or TIREM4) resulted in an average simulation runtime improvement of 14% relative to the PLM methodology. This was surprising since one of the drivers in developing the PLM method several years ago was to improve simulation runtime[3]. However, upon examining the OPNET 11.5 TIREM-generated results in greater detail, as mentioned earlier, we identified the efficiency threshold parameter implemented by OPNET to improve simulation runtime which it does. However, care must be taken to verify that the value assigned in OPNET is appropriate for the specific scenario(s) being investigated since it is not a user-assigned parameter and is transparent to the user during simulation set-up.

Results were also generated comparing the runtime performance impact of TIREM3 to TIREM4, both using OPNET 11.5. The runtime performance improvement observed with TIREM4 was negligible compared to TIREM3.

Finally, we observed a simulation runtime performance improvement of up to 24% when comparing the TIREM3 OPNET 11.5 results to the TIREM3 OPNET 11.0 results.

In conclusion, from a simulation runtime performance standpoint, the results indicate that the user should run either TIREM3 or TIREM4, inline (i.e., as the OPNET simulation is running) using OPNET 11.5 in order to achieve the best runtime performance when terrain must be accounted for in the path attenuation calculations.

VIII. PARALLEL PROCESSING

We are investigating the achievable gain in simulation runtime using parallel processing. Unfortunately, few existing OPNET process models are designed to support multi-threaded processing using a multi-processor computer system. Nevertheless, since our profiling showed about 50% of the run time is involved in computation of forwarding tables by OLSR, we decided to run all modules sequentially except for MANET Route Manager and OLSR. Only these two process models are configured to use multi-threaded processing to take advantage of available parallel processing hardware.

These process models were initially designed for sequential processing only. To support multi-threaded processing, these two process models were enhanced by employing a serializing mechanism to ensure critical sections of the code are not run concurrently.

There are several code sections of MANET Route Manager and OLSR that should not run concurrently. These sections deal with creation of a child process, creation of packets and data structures, and any other operation that involves changes to common memory locations. For this type of code that should not be executed concurrently, we used the mutual exclusion (mutex) mechanism to serialize the code execution and to ensure that at most one thread accesses the code at a time.

When multiple threads running on multiple processors are trying to access a section of critical code, mutexes ensure only one will be granted permission to proceed. The other threads must wait for the active thread to complete the execution before they can proceed.

OPNET uses a “parallel event execution time window” during which events are considered independent from each other and therefore can be parallelized and be assigned to separate CPUs. This window should be small enough to ensure parallelized events can be executed on multiple CPUs without affecting the final simulation results.

We experimented with several time window values expecting the parallel simulation run time to be much lower than the run time for a sequential simulation run for which all the process models are configured to run sequentially on one CPU. Unfortunately, for every parallel simulation run that produced correct results, its run time was slightly higher than the run time for an equivalent sequential run.

[3] The other driver was to provide a mechanism with which to apply TIREM to OPNET simulation for the calculation of path attenuation. At the time that the PLM methodology was developed, OPNET did not support a direct link with TIREM.

IX. CLOSING REMARKS

The purpose of this paper was to report on the results of our investigations to improve simulation runtime performance of mobile ad-hoc communication networks. We investigated a number of existing features available in OPNET including Packet Copy, Packet Duplication, Wireless Domain, Dynamic Receiver Group, and Hybrid Simulation. Unfortunately, none of these techniques significantly improved runtime performance due to two primary reasons: (1) they were not applicable to wireless applications or (2) they were developed to improve runtime perform via optimizations in the OPNET radio pipeline which, through detailed profiling analyses, was shown to only contribute less than 3% toward the total simulation runtime for our mobile, wireless scenario.

We then extended our profiling analyses and identified the important result that the route formation calculations contributed to more than 50% of the total runtime and, more specifically, OLSR's proactive routing modules were the top consumers of simulation runtime. We then focused our effort toward improving the route formation algorithms and achieved some success in reducing simulation runtime - on the order of 27% improvement.

In related efforts, we are investigating the utility of parallel processing and HLA techniques to improve runtime performance; however, results to date indicate that these techniques will only achieve limited success due to the nature of mobile, wireless networks which, using existing techniques, precludes their partitioning onto separate processing platforms and/or processors for all but a very limited family of scenarios. In closing, we recommend that future research be focused on this specific area. That is, the development of procedures and algorithms for partitioning mobile, wireless networks such that the use of multiple platforms via co-simulation can be applied to significantly reduce simulation runtime in Mobile Ad-Hoc Network (MANET) applications.

REFERENCES

[1] Comparetto, G., Lindy, E., Mirhakkak, M., and Schult, N., *"Overview and Application of a Modeling and Simulation Environment to Support Protocol Performance Evaluations in Mobile Communications Networks"*, presented at the 2004 International Conference on Modeling, Simulation and Visualization Methods (MSV'04), Las Vegas, NV, Paper # CIC2471, 21-24 June 2004.

[2] Comparetto, G., Schult, N., Mirhakkak, M., Chen, L., Wade, R., Duffalo, S., *"An End-to-End Modeling and Simulation Testbed (EMAST) to Support Detailed Quantitative Evaluations of GIG Transport Services"*, accepted for presentation at the 10th International Command and Control Research and Technology Symposium (ICCRTS), McLean, VA, 13-16 June 2005.

[3] T. Clausen (ed) and P.J acquet (ed).*Optimized link state routing protocol (OLSR),* October 2003.RF C 3626.

[4] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei, *Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification*, June 1998, RFC 2362.

[5] *"Improving Runtime and Memory Footprint Performance in Large Scale Network Simulations"*, G. Comparetto, et. al., MILCOM 2005, Atlantic City, NJ, October 17–20, 2005

[6] "*Optimizing Route Formation Algorithm to Reduce Simulation Run-Time for Large Tactical Networks*", M. Mirhakkak, et. al., Scheduled for presentation at MILCOM 2006, Washington DC, October 23-26, 2006