

# PERFORMANCE ANALYSIS OF DRAMA: A DISTRIBUTED POLICY-BASED SYSTEM FOR MANET MANAGEMENT

Cho-Yu Jason Chiang, Stephanie Demers, Praveen Gopalakrishnan, Latha Kant, Alex Poylisher, Yuu-Heng Cheng, Ritu Chadha, Gary Levin, Shihwei Li, and Yibei Ling  
Telcordia Technologies, Piscataway, NJ

Scott Newman, Lorraine LaVergne and Richard Lo  
U.S. Army CERDEC, Fort Monmouth, NJ

## ABSTRACT

*DRAMA<sup>1</sup> is a distributed policy-based management system designed to manage Mobile Ad hoc NETWORKS (MANETs). Its design philosophy is to create intelligent, self-adaptive policy agents to manage dynamic networks without human intervention. Network management functions are performed in a distributed fashion by these policy agents, rather than being controlled by a centralized management station. Policies are used to control the frequency and content of network management messages exchanged among policy agents in a way that reduces bandwidth usage and increases the utility of management messages. This greatly enhances management efficiency and reduces the bandwidth overhead required for network management. As with any new technology, there is a question about the scalability of this approach. The purpose of the work described in this paper is to study whether the DRAMA policy-based network management system can scale to networks of 500+ nodes. The study uses a novel simulation-based approach to evaluate DRAMA performance when DRAMA is used to manage MANETs of up to 500 nodes. The results confirm that the DRAMA distributed policy-based management paradigm provides superior performance over a centralized management paradigm for MANETs.*

## INTRODUCTION

Mobile Ad hoc NETWORKS (MANETs) are the building blocks of the U.S. Army FCS networks [1]. The distinguishing characteristics of MANETs include absence of any infrastructure and dynamic network topology. MANET nodes are mobile routers that use radios to communicate wirelessly. Because physical environments are noisy and distances between nodes change over time, the bandwidth on radio links fluctuates and link connectivity is unstable. Past research on MANETs has been focused mostly on enabling technologies such as

radios, media access, routing, etc. With the coming of age of these networks, a grand challenge has emerged: how can MANETs be managed efficiently and effectively in dynamic and bandwidth-constrained environments?

Today's network management tools were designed for the bandwidth-abundant Internet with stable connectivity. These tools are ill-suited for military MANETs due to the differences in characteristics between the Internet and such networks. The Internet is connected by fibers and cables that supply abundant and stable bandwidth; its operations depend on static, human-administered configurations; and management systems do not need to limit the amount of bandwidth consumed by their own traffic, since bandwidth is abundant. The common Internet management model is based on the manual setup of a static management hierarchy, and management systems do not adapt their behavior based on network changes. In sharp contrast, systems managing MANET must possess self-adaptive and self-healing capabilities because network conditions may change significantly at any time. In addition, they must regulate network management traffic to conserve bandwidth.

Dynamic Readdressing And Management for the Army (DRAMA) [4] was a Science and Technologies Objective of the U.S. Army CERDEC. The DRAMA STO was launched to develop a distributed management system suitable for managing MANETs. One of the exit criteria for the DRAMA STO was to verify that the management system could manage MANETs of 500 nodes. The implications of this requirement are two-fold: the first is to verify that the distributed network management approach will scale to MANETs of 500 nodes; and the second is to verify that this approach will enhance network management performance, as compared to that of the conventional SNMP-based approach. This paper reports the use of a novel simulation-based approach that uses the actual DRAMA software in conjunction with a simulated network to evaluate DRAMA performance in MANETs of up to 500 nodes.

The rest of this paper is organized as follows. First we briefly describe the architecture of the DRAMA system, restricting our focus to the autonomous formation of a

<sup>1</sup> The research reported in this document/presentation was performed in connection with contract number DAAD19-01-C-0062 with the U.S. Army Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as presenting the official policies or position, either expressed or implied, of the U.S. Army Research Laboratory, or the U.S. Government unless so designated by other authorized documents. Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

management hierarchy. Next we examine the conventional modeling and simulation performance analysis approach, articulate why we need to develop a hybrid simulation approach to evaluate the performance of the DRAMA system, and explain at a high level how our approach works. We then describe the experiment setup, present our analysis of the simulation outcome, and discuss possible future work.

### SYSTEM ARCHITECTURE

The high-level architecture of the DRAMA management system is shown in Figure 1. As shown, a collection of *Policy Agents* [2][3] with different roles are used to manage a MANET. At the highest level, the *Global Policy Agent*, or *GPA*, manages multiple *Domain Policy Agents*, or *DPAs*. A *DPA* can manage multiple *DPAs* or *Local Policy Agents (LPAs)*. An *LPA* manages a node, and can also manage nearby devices. *LPAs* perform local policy-controlled configuration, monitoring, filtering, aggregation, and reporting, thus reducing management bandwidth overhead. Policies are disseminated from the *GPA* to *DPAs*, and then from *DPAs* to other *DPAs* or *LPAs*. Policy agents react to network status changes on various levels (globally, locally, domain-wide) by automatically reconfiguring the network as needed to deal with fault and performance issues. In this architecture, any node can dynamically take over the role of another node, which ensures system resilience. The flexible policy agent framework allows adjustment of management functionalities and behaviors via policy changes.

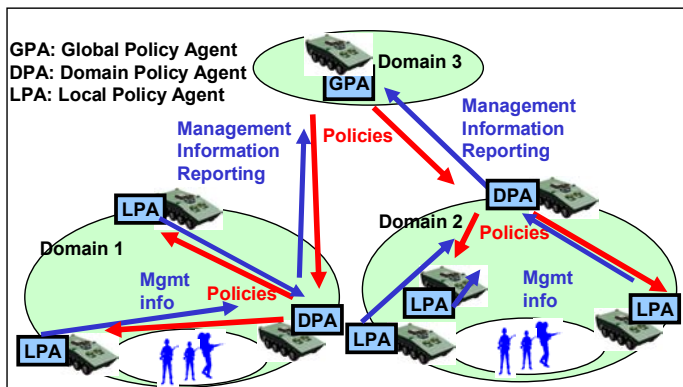


Figure 1 The high-level architecture of the DRAMA system

Given that a multi-tier hierarchy is essential for scaling a distributed management system, it is critical for this hierarchy to systematically and autonomously self-form, especially when the system will be used to manage MANETs. MANET nodes may lose network connectivity temporarily or permanently. As a result, the management hierarchy will require adjustments from time to time. To address this issue, the DRAMA system enables nodes to

form “clusters” and then link clusters into a tree-like hierarchy. A cluster includes a single cluster leader and a set of cluster associates. Leaders of several clusters can form another cluster, and this process can be repeated to form a hierarchy comprising a cluster of recursively-formed clusters. The DRAMA system allows a hierarchy to have an arbitrary depth, since limiting the depth of the hierarchy will affect the scalability of the system; it can also adjust the hierarchy to the size of the network by expanding or shrinking the number of tiers in the hierarchy. A sample DRAMA management hierarchy is shown in Figure 2. More details about the DRAMA system are described in [3][4].

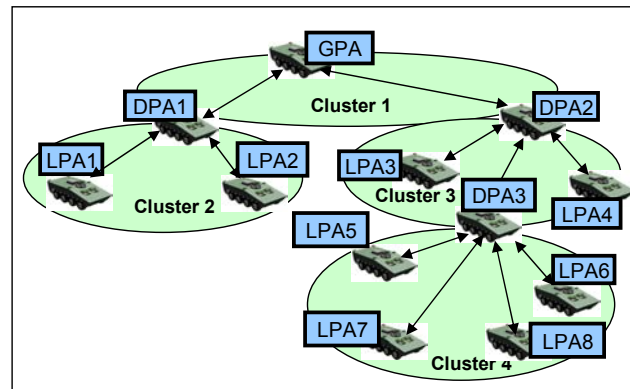


Figure 2 A sample DRAMA management hierarchy

### INITIAL HIERARCHY FORMATION

The initial hierarchy formation is guided by advance planning. The plan can mandate that the management hierarchy map to the human reporting structure to handle management needs originating from human administration, or that the management hierarchy approximate the packet routing structure to reduce bandwidth consumption by network management traffic. Both approaches are supported by the DRAMA system.

In this study, a pre-planned management hierarchy based on the human reporting structure is used as the initial configuration, which is referenced by all the nodes to bootstrap the setup of a hierarchy. Every node is given the planned hierarchy a priori. A node initializes itself by contacting its default parent node according to the plan to form a parent-child relationship. If this attempt fails, it will try to register itself as a child node with the other parent node candidates, which include the ancestor nodes of its default parent node all the way up to the root node in the hierarchy. In case a node fails to register with a parent node via the above process, it will temporarily become a *GPA*, and then repeat the above process all over again until it successfully registers itself with a parent node.

## CLUSTER MAINTENANCE

After the initial formation of a management hierarchy, it is necessary to maintain this hierarchy for the purposes of distributing policies and collecting management information. The cluster maintenance procedure is described as follows. First, the leader of a cluster periodically sends heartbeat messages to all its associates. These heartbeat messages contain information about the cluster leader and cluster settings. A cluster associate acknowledges receipt of heartbeat messages from its leader in order to maintain its membership in the cluster. If a cluster associate misses a number of consecutive announcement messages, it assumes that its leader is not reachable and starts the parent node search process, as it has done in the initialization phase. Further, if a cluster becomes “too big” (where “too big” is a configurable metric), it splits into two or more clusters.

For cluster signaling, two types of messages are used. When a node wants to join a cluster (when two clusters are merging), it sends a “*join request*” message to the cluster leader. If the request is granted, the leader sends a “*join granted*” message to the child and records the node as its child when an acknowledgement message is received. When a cluster leader splits its cluster into two or more clusters, it appoints one of its associates as the leader for a group of associates in its cluster. The new leader remains a child in its current cluster, while the associates to be split out join the new cluster by signaling their new leader.

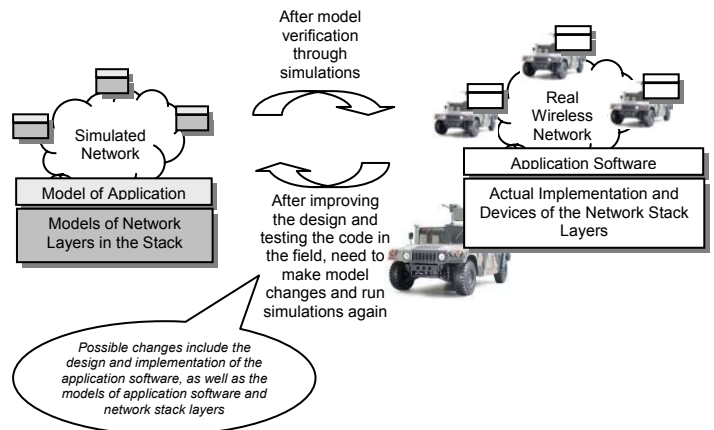
## PERFORMANCE ANALYSIS CONSIDERATIONS

Maintaining a management hierarchy incurs bandwidth overhead. It is important to understand quantitatively the overhead as a function of the network size in order to evaluate scalability. Typically such an evaluation is performed by conducting a simulation study. However, the conventional *Modeling and Simulation (M&S)* approach faces challenges when it is applied to analyzing the performance of MANET software, because ensuring modeling and simulation fidelity becomes an issue. Another issue is that the ability to conduct a simulation study for MANETs in the scale of 500 nodes is limited by the simulation software and hardware capacity. We will elaborate on both issues further below.

## MODELING AND SIMULATION FIDELITY

The issue of modeling and simulation fidelity arises mainly due to the lack of mature MANET software design paradigms. In current practice, the design of MANET software typically needs to be altered multiple times at the software testing phase, thus making it difficult to maintain the consistency between the software and its simulation model. This issue arises mainly because the characteristics of MANETs require fundamental software design

paradigm shifts. For example, “no route to host” should no longer be considered an exception by MANET software. Moreover, the network application API was originally designed for the Internet software. This API in many ways is not suitable for developing MANET software. The current practice depends on making incremental enhancement to the software design as new issues arise during the lab testing and field testing phases. Consequently, simulation models cannot make over-simplifying assumptions about the modeled software, and the models must closely approximate the new software design paradigms. However, the inability to properly specify the design of complex MANET software inevitably causes software design changes late in the development cycle. To ensure fidelity of the results of a simulation study, we must maintain the consistency between software and its simulation model. These iterations between MANET software changes and the corresponding model changes are illustrated in Figure 3.



**Figure 3 Iterations of Changes between Software and its Model for Tactical Networks**

## SIMULATION SOFTWARE/HARDWARE CAPACITY

The other major issue with the conventional M&S approach is that when conducting a large-scale high-fidelity MANET simulation study, the capacity of the simulation software and hardware limits the applicability of M&S. Even though parallel simulation is a powerful tool for running large-scale simulations, as of today most simulation software does not fully support it since it requires rewriting most of the existing code to enable parallelism. On the other hand, simulation software that supports parallel simulation typically does not possess a rich set of features to support various simulation needs; the degree of simulation parallelism that can be achieved also has a limit and hence the actual performance enhancement when simulation is distributed over multiple CPUs scales poorly with the number of CPUs being used.

Second, simulation software by itself is an operating platform. It may not have been adequately optimized in terms of resource utilization that is necessary for enhancing the efficiency of large-scale simulations. Such simulations use a large number of modules and processes, and take a long time to complete. Process scheduling and memory allocation can significantly affect the time needed to complete simulations. Inefficient process scheduling will result in overuse of CPU cycles and inappropriate memory allocation will result in excessive page faults. Both factors will significantly prolong the time needed for running large-scale simulations.

Third, large-scale simulations are common practice today; however, most of them are performed on super computers for efficiency reasons. Running large-scale network simulations on super computers may not be cost-effective.

In summary, running large-scale simulations is both time-consuming and expensive. It can take days or even weeks merely to run a few minutes of simulation. Therefore, a better approach that speeds up large-scale MANET simulations is needed.

### HYBRID SIMULATION

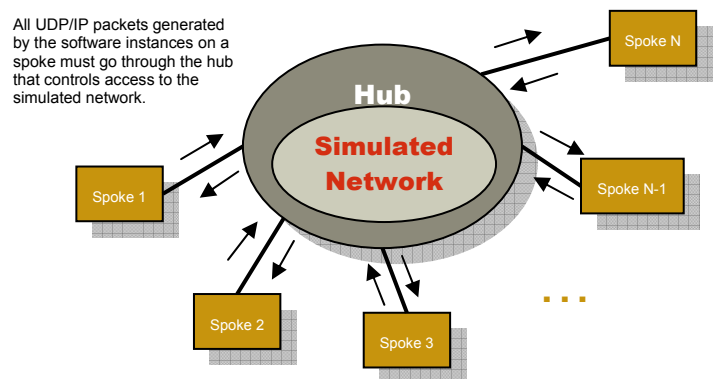
Software-in-the-loop (*SITL*) simulation [8] is the basis of the hybrid simulation approach that was used in our study. The idea is simple—using software itself rather than its model in simulations. When running SITL simulations, software instances will be standalone processes rather than threads or objects managed by the simulation software, often called the simulator. Software instances participate in simulations through invocations and callback registrations on the simulator’s co-simulation API to coordinate their executions with the simulator. When simulating distributed software in a large network, typically a large number of software instances are hosted by machines other than the one hosting the simulator to reap performance gain from computation distribution. Such a methodology is especially attractive while analyzing large-scale ad hoc networks, as in our case, where there do not exist formal specifications of the complex network-based software. Further, such a methodology provides invaluable information about the scalability and performance of “the actual software implementation” rather than a model of the software.

Our goal was to use software as is in large-scale simulations, which, as our research has shown, is possible if the development of software follows good programming practice. By using software rather than its model in simulations, we completely avoid building software models and can focus on setting up simulation scenarios to properly establish scalability arguments. More specifically, we reap the following benefits with this approach:

- *Achieve the highest possible simulation fidelity.* The SITL simulation approach provides the highest possible fidelity in simulating network-based software performance. The SITL simulation approach completely eliminates the need for developing models for simulation. The use of the same code for both field executions and simulations means that the problem of the software model getting out of sync with the software cannot arise.
- *Reduce time needed for running large-scale simulations.* The SITL simulation approach intrinsically distributes the computational load from one CPU to multiple CPUs. The computation load on the CPU running the simulator is reduced, thus reducing time needed for running large-scale simulations.
- *Perform more rigorous test of software.* By directly plugging software into the simulation framework, we can perform more rigorous test of software before using it in field trials. This approach provides an economical and effective way of performing multi-unit testing and integration testing of software. Therefore, the SITL simulation approach can help reduce the amount of time needed for field trials, which is invaluable as large-scale field trials pose heavy demands in terms of hardware assets as well as personnel support.

### SOLUTION ARCHITECTURE

In this section, we give a high-level overview of the solution architecture. The details of this innovation are described in [6][9]. The key concept of our approach is in the use of a simulated MANET that *virtually* connects the actual software instances. From the software execution viewpoint, the software cannot distinguish such a virtual network from a real one. The virtual network, as powered by the network simulator, aims at simulating all activities and their resulting consequences that would occur when packets traverse a real network.



**Figure 4 The Hub and Spokes Architecture of the Hybrid Simulation Framework**

Figure 4 illustrates the hub and spoke architecture of this solution. Software instances are spokes and exchange messages with each other via the hub. The hub hosts the simulated network, which forwards and drops packets according to the simulation. Packets will be forwarded if at the time they arrive at the hub, end-to-end connectivity exist between the originating spokes and the destination spokes in the simulated network; otherwise they will be dropped.

In the context of this study, the spokes are DRAMA software instances that are hosted by an array of Linux machines. The spokes do not exchange any message directly with one another since all traffic must transit through the hub. Therefore, in the hub the “shell” surrounding the simulated network is responsible for transparently forwarding packets from the DRAMA software instances to the simulated network, and from the simulated network to the DRAMA software instances. This shell is the most essential component in our SITL simulation solution framework. The simulated network can be provided by any network simulator, as long as the two-way packet forwarding between the shell and the network simulator is feasible. This solution architecture is independent of the choice of network simulators, as long as the simulator supports the above concept. In this study, we used OPNET [7] as the network simulator.

Note that the architecture shown in Figure 4 was presented in an abstract sense. In actual implementation, the simulated network is powered by OPNET on a single CPU. DRAMA software instances on the “spokes” are hosted by other CPUs. The shell bridging the DRAMA software instances and OPNET consequently has to split into two parts—one is called “Co-Simulator”, which is co-located with OPNET; and the other is called “Packet Mangler”, which runs on every machine hosting the DRAMA software instances.

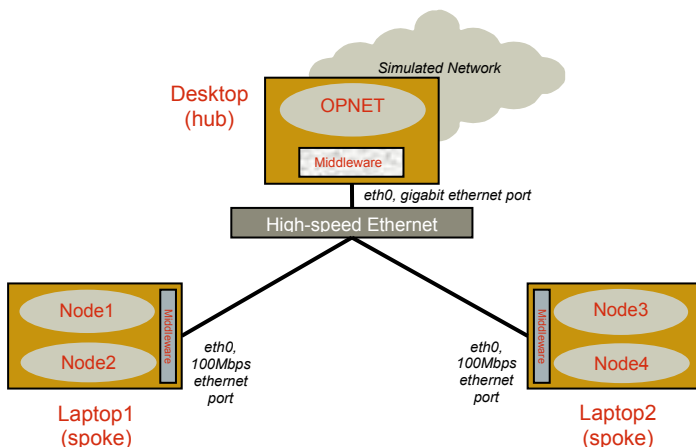


Figure 5 An illustration of the SITL Simulation Architecture

We use a simple example as shown in Figure 5 to illustrate our implementation of the hub and spokes hybrid simulation architecture. In this example, all three machines, including two laptops and one desktop, are physically connected to a high-speed Ethernet switch. The laptops run two DRAMA software instances each and the desktop runs OPNET that provides a simulated network. In this example, a software instance is denoted by *NodeX*, where  $1 \leq X \leq 4$ . On both laptops and the desktop, the middleware components function as packet redirectors. On the laptops, only traffic generated by DRAMA software instances will be intercepted and redirected to the desktop; other regular protocol stack traffic such as those from ARP and RARP will not be redirected. The middleware component on the desktop is responsible for injecting packets into the simulated network when it receives them from the laptops, and for forwarding the packets from the simulated network to the machines hosting the intended DRAMA recipients.

### TEST BED SETUP

The DRAMA simulation test bed includes an enterprise server running Microsoft Windows Server 2003 Standard Edition, 24 PCs (referred to as pods) running Linux 2.4.22 kernel, and a fast Ethernet switch. The server is an HP machine with two 3.4GHz Intel CPU and 3.5GB memory. The other PCs are Pentium III 700 MHz machines equipped with 768MB memory. The enterprise server hosts the OPNET simulator. DRAMA software instances (also referred to as DRAMA nodes) run on the pods, and each pod can run multiple DRAMA nodes simultaneously. Packets from one DRAMA node destined for another DRAMA node, regardless of their hosting pods, must always transit through the network simulator. To ensure that no packet is dropped due to bandwidth limitations and that packet forwarding delay is negligible between the pods and the OPNET server, we use a Cisco Catalyst 2950 fast Ethernet switch with one of its gigabit ports connecting to the server hosting the OPNET. All the machines are configured to be on the same subnet 192.168.99.0. The OPNET server is configured with an IP address of 192.168.99.254. The rest of the machines are numbered sequentially: Pod1’s IP address is 192.168.99.1, Pod2’s IP address is 192.168.99.2, and so on.

### EXPERIMENTS AND ANALYSIS

Recall that this scalability study has two objectives. The first is to verify that the agent-based distributed network management approach can scale to MANETs of 500 nodes. The second objective is to verify that this approach greatly enhances network management performance. In essence, the focus of the first objective is on studying the relationship between the number of nodes and the bandwidth overhead resulting from maintaining a management hierarchy for all nodes; the second objective

is to compare the performance of the distributed management approach adopted by the DRAMA system with that of the existing SNMP-based approach, verifying that substantial performance enhancement will be achieved.

We conducted experiments with two different focuses. The first focus of the study was on the scalability aspect of the DRAMA software, which hinges on the dynamic clustering mechanism that forms and maintains the network management hierarchy. Note that in this scalability verification study we measured scalability in terms of clustering message bandwidth overhead with respect to the number of nodes. We set the number of nodes to 120 (5 DRAMA instances per pod), 240 (10 DRAMA instances per pod), and 504 (21 DRAMA instances per pod) for three scenarios, respectively. All three scenarios used comparable mobility patterns, which showed group mobility trajectories as nodes were divided into correlated moving groups. Most scenarios were run three times with different random number seeds to account for simulation variability, except for the 504-node scenario. Each scenario was run for 5400 seconds to weed out transient effects and to ensure that the run had stabilized and the collected results were valid. Again one exception to the simulation duration was made for the 504-node scenario, which was only run for 3726 seconds since seven days had already elapsed before we terminated that run.

The second focus of the study was to compare the performance of the DRAMA network management paradigm with that of the SNMP-based network management paradigm. The former features a dynamic, distributed, policy-controlled mechanism that proactively pushes network management data up the hierarchy, while the latter depends on a centralized network management station that regularly polls all the nodes to collect data. Assume that policies are distributed once as an initialization step. Aside from clustering traffic, DRAMA network management traffic mainly consists of the data reporting traffic that is sent by every single node to its parent node in the hierarchy (will be referred to as *up* traffic). Whether a parent node forwards up the received reports depends on the reporting/filtering policies in use. In this study, we modeled the behaviors of implementing different reporting policies by using different report forwarding probability. The rationale behind filtering reports via policies is that (i) reported data may not have changed much as compared to that in the previous report, and (ii) reported data may have little value from the network management viewpoint. All DPAs (intermediary nodes) used the same forwarding probability. The range of the probability is between 0 and 1, where 1 indicates that the DPAs must always forward all the reports they have received to their parent nodes, modeling the case where there is no filtering policy. A forwarding probability less

than 1 indicates that report filtering policies are being implemented by the DPAs. We used 1.0, 0.5 and 0.1 as the forwarding probability in three experiment scenarios, respectively.

To compare the results of the above experiments with of the results obtained by using SNMP to manage the same network, we modeled SNMP management behavior using a static two-tier hierarchy which had one network manager sending SNMP query messages to all the other nodes periodically. When nodes received the SNMP query messages, they would respond by sending SNMP response messages to the network manager. Performance statistics collected for the scenarios include the total messaging bandwidth overhead and message delivery ratio (MANET connectivity is not stable). When comparing the performance of DRAMA approach with that of an SNMP-based approach, we used 120 mobile nodes in all scenarios. The same mobility pattern was used for all runs. Each scenario was run three times and each run lasted 3600 seconds.

### SCALABILITY ARGUMENT

First we examine the relationship between the DRAMA traffic for dynamic clustering maintenance purposes and the number of nodes in the simulated network. As shown in Figure 6, with data points derived from the scenarios for 120 nodes, 240 nodes, and 504 nodes, we obtain two lines depicting the growth relationship between the X and Y axes. We observe that there is slightly more DRAMA clustering traffic sent than DRAMA clustering traffic received. This is because some of the sent packets are dropped in the simulation. Since both curves are roughly linear as expected, we confirm that the DRAMA clustering traffic will not grow exponentially as the size of the network increases.

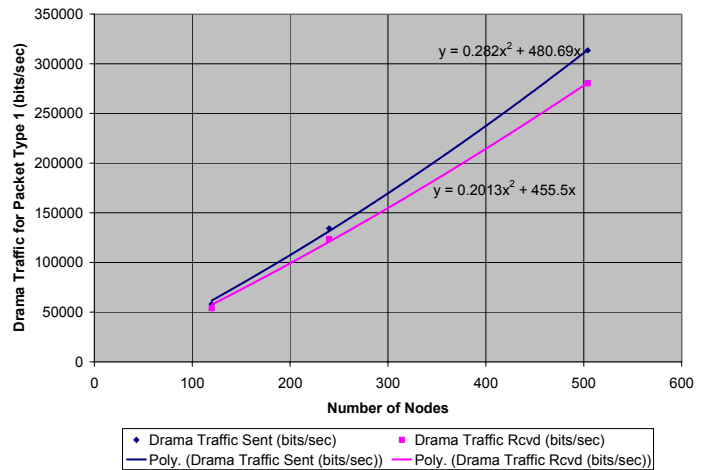


Figure 6 Clustering Traffic Sent and Received in bits/sec

Next we examine the performance of the DRAMA approach as compared to the SNMP approach. Here we denote the data derived from running the SNMP scenario as baseline. The first point of interest is the average number of hops that a packet needs to traverse in all scenarios. This statistic gives an indication of the bandwidth consumption, if all scenarios generate the same number of packets. As seen in Table 1, the average number of hops in the baseline scenario was higher than those in the others. This is because the reporting hierarchy is loosely related to the routing structure in the mobility profile and thus the number of IP hops is lower. We believe that this is a reasonable assumption reflecting the typical military force deployment. If the reporting structure and management hierarchy is more closely aligned to the routing structure, as might be the case, we expect to see lower numbers in the “Report Up” columns than the number in the “Baseline” column as shown in Table 1.

**Table 1 # Hops Comparison between DRAMA and SNMP**

	Report Up			Baseline
	Fwd 1.0	Fwd 0.5	Fwd 0.1	
# IP Hops	5.04	4.80	4.92	7.68

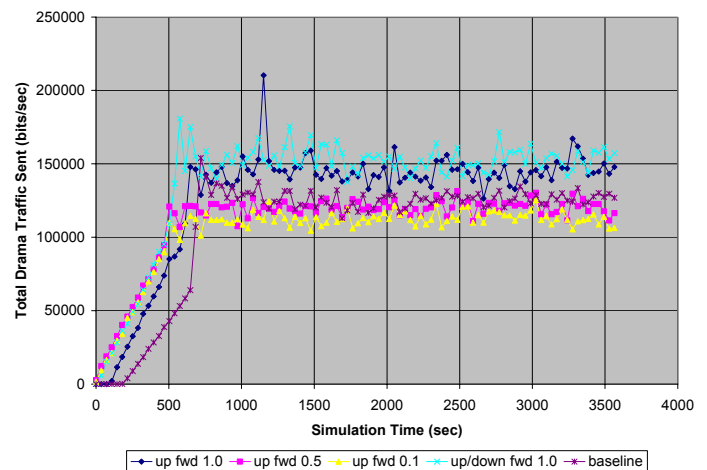
### PERFORMANCE COMPARISON

Aside from the clustering traffic, DRAMA network management traffic includes infrequent *down* traffic (1024 bytes sent once every 600 seconds) and regular *up* traffic (512 bytes sent once every 30 seconds). A parent node may forward the reports received from its children nodes to its parent node, depending on the forwarding probability. The forwarding probability is controlled to study the filtering effect of policy control. We used 1.0, 0.5 and 0.1 as forwarding probability in our scenarios, respectively. Note that forwarding probability was only applied to the intermediate nodes (i.e., DPAs) and was not applied to the reporting by the leaf nodes (i.e., LPAs).

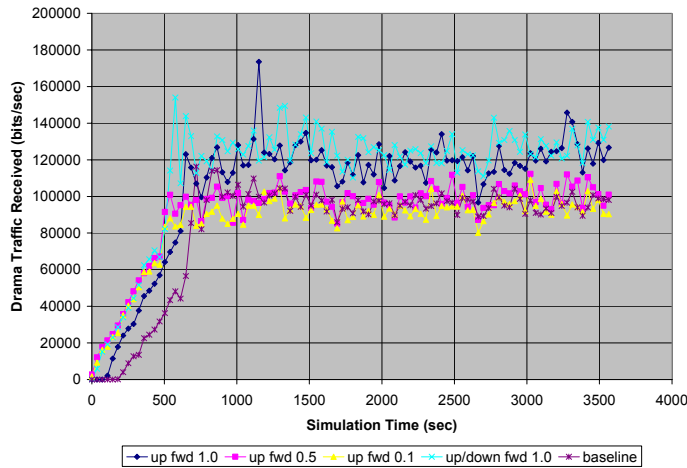
Below we first describe the rest of the instrumentations common to all the simulation scenarios used in this study. The number of nodes in the network is fixed at 120. All 120 nodes are moving as one single group. With the mobility model we used, a node could lose IP connectivity to its preferred parent node roughly 10% of the time.

There is little difference in the DRAMA traffic sent and received when the forwarding probability is between 10% and 50%, while there is a significant difference when 100% is used as the forwarding probability, as shown in Figure 7 and Figure 8. Note that the amount of traffic reduced by decreasing the forwarding probability will not be linear. Assume that we have a hypothetical 3-tier network with a root at the top of hierarchy, 10 nodes

immediately below the root as the intermediate nodes, and 100 more nodes at the bottom of this hierarchy. With 100% forwarding probability, there will be 210 packets sent in total. This is because the bottom tier collectively sends up 100 packets and the 2nd tier collectively sends up their own 10 packets and the 100 packets received from the 3rd tier. With 10% forwarding probability, there will be only 120 packets sent in total. The 3rd tier nodes send up 100 packets and the 2nd tier nodes only send up 10% of these 100 packets plus their own 10 packets for a total of 120 packets. Therefore, in this example, the total traffic sent up is equal to 210, 160, and 120 packets for the 100%, 50%, and 10% forwarding probability scenarios, respectively. This corresponds to 100%, 76.2%, and 57.1% of the total traffic received for the 100%, 50%, and 10% forwarding probability scenarios respectively with a difference in percent traffic received of 23.8% between the 100% and 50% case and 19.1% between the 50% and 10% case. As the number of tiers increases, this difference is more apparent. For a hypothetical 4-tier hierarchy with each node having 10 children nodes, the total traffic sent up is equal to 3210, 1910, and 1230 for the 100%, 50%, and 10% forwarding probability scenarios respectively. This corresponds to 100%, 59.9%, and 38.3% of the total traffic received for the 100%, 50%, and 10% forwarding probability scenarios, respectively. In the simulation scenarios, the number of tiers is typically between 3 and 4. Consequently, this explains why the results obtained from the 10% and 50% forwarding probability scenarios are much closer when they are shown alongside the results obtained from the 100% forwarding probability scenario.



**Figure 7 DRAMA and SNMP management traffic sent**



**Figure 8 DRAMA and SNMP management traffic received**

The baseline run represents the SNMP-based network management paradigm. In the baseline run, node 1 sends queries to the rest of the 119 nodes once every 30 seconds. As shown in Figure 7 and Figure 8, the total DRAMA traffic sent and received in the baseline run compares to the total DRAMA traffic sent and received in the up scenario with a forwarding probability of 50%. However, if LPAs send their reports according to the forwarding probability as well (which is possible and reasonable with the presence of filtering policies in place), we expect to see more substantial performance gain by DRAMA, as compared to SNMP.

Lastly, the other subject of interest is the timeliness benefit of employing a dynamic network management hierarchy. Thanks to the inherent store-and-forward management information report structure, a self-healing management hierarchy can reduce the time needed for a report to reach the top of the hierarchy in MANET environments. It is important to measure this time because global management decisions may have to be made by personnel at the top of the hierarchy. With a network management hierarchy, the information is typically reported by a node to its parent node. The parent node decides whether to forward any information it receives from its children nodes further up the hierarchy along with its own information. Conceptually, the closer the clustering structure maps to the network topology, the “closer” (in terms of number of IP hops) information will move to the top of the hierarchy at each step of the report forwarding process. In MANETs the wireless links can be very unstable; thus the fewer the number of hops between a sender and a receiver, the higher the probability that a message from the sender can reach the receiver. If a parent node is typically on the path from a node to the centralized manager node, useful information will have a better chance of being delivered to the top of the hierarchy in a timely manner. From Table 2

we can see that our simulation setups have been designed to align with the hierarchy with the network topology, although they are not identical.

**Table 2 The Packet Delivery Ratio Comparison**

# Msgs=12000	Report Up			Baseline
	Fwd 1.0	Fwd 0.5	Fwd 0.1	
# Delivered Msgs	7601	7281	6947	3015
Packet Delivery Ratio	63%	61%	58%	25%

## SUMMARY

In this paper, we presented a study of the scalability and performance of the DRAMA management system. Our analysis of the simulation results shows that the DRAMA system can manage MANETs of 500+ nodes while the incurred bandwidth overhead grows linearly with the size of the network. The analysis also shows that the performance of the DRAMA system in terms of reporting timeliness and message delivery ratio is much superior to that of conventional SNMP-based network management systems.

## REFERENCES

- [1] US Army, “Future Combat Systems”, <http://www.army.mil/fcs/>
- [2] J. Ferber, “Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence,” Addison Wesley, 1999.
- [3] C. Chiang, R. Chadha, Y-H Cheng, S. Li, G. Levin, and A. Poylisher, “A Novel Software Agent Framework with Embedded Policy Control”, Proceedings of the Military Communications Conference (MILCOM 2005), Atlantic City, NJ, Oct. 17-20, 2005.
- [4] R. Chadha, Y-H Cheng, C. Chiang, S. Li, G. Levin, and A. Poylisher, “DRAMA: A Distributed Policy-Based Mobile Ad Hoc Network Management System”, Proceedings of the Military Communications Conference (MILCOM 2005), Atlantic City, NJ, Oct. 17-20, 2005.
- [5] R. Chadha, Y.-H. Cheng, C.-Y. J. Chiang, G. Levin, S. Li, and A. Poylisher, “DRAMA: A Distributed Policy-based Management System”, The Third International Conference on Mobile Systems, Applications, and Services, June 6-8, 2005, Seattle, WA.
- [6] R. Chadha et al., “DRAMA Performance and Scalability Analysis Report”, a deliverable to U.S. Army CERDEC, December 2005.
- [7] OPNET, [www.opnet.com](http://www.opnet.com).
- [8] P. Riley and G. Riley, “SPADES--A Distributed Agent Simulation Environment with Software-in-the-Loop Execution”, In *Winter Simulation Conference Proceedings*, pp. 817–825, 2003.
- [9] C. Chiang et al., “A Novel Hybrid Simulation Approach for Software Correctness Verification and Performance Evaluation”, to be submitted for publication.