VIRTUAL AD HOC NETWORK TESTBEDS FOR HIGH FIDELITY TESTING OF TACTICAL NETWORK APPLICATIONS

Alex Poylisher, Constantin Serban, John Lee, Ted Lu, Ritu Chadha, Cho-Yu J. Chiang
Applied Research, Telcordia Technologies
{sher, serban, jolee, tedlu, chadha, chiang}@research.telcordia.com

Kimberly Jakubowski, Robert Orlando
U.S. Army CERDEC
{Robert.Orlando1, Kim.Jakubowski}@us.army.mil

ABSTRACT

Testing of applications for tactical MANETs poses a special technical challenge due to the difficulty of conducting experiments in an ad hoc network environment at a scale larger than a few nodes. One approach is to conduct experiments in a testbed that can imitate a tactical MANET to the highest feasible level of fidelity. For applications, this is achieved by executing unmodified software under real operating systems and realistic hardware resources. For the network, it is achieved by using an emulated/simulated network that behaves like a real MANET in some/all layers of the protocol stack.

In this paper, we describe the current state of Virtual Ad hoc Network (VAN) Testbed[1] technologies for constructing testbeds that allow testing unmodified tactical applications over simulated MANETs, including network-aware applications. On the application side, we employ Xen-based virtualization to provide high fildelity of application execution environment, testbed resource scalability and manageability. On the network side, we employ user-provided simulation models and enable seamless integration of the hosts that run the software under test with virtual nodes in a simulated network. Our goal is to enable running experiments over large-scale (500-1000 nodes) MANETs using VAN testbeds.

1. INTRODUCTION

Mobile Ad hoc NETworks (MANETs) are characterized by their capability to enable networking without any infrastructure support. Specifically, there are no wires and no dedicated routers. As there are no wires, nodes use wireless radios with limited bandwidth and typically high loss rates to communicate; as there are no dedicated routers, every node participates in packet forwarding; as the location of a node is not fixed and nodes can enter and leave the network at any time, network topology is dynamic.

Testing and evaluation of the functional correctness and communications performance of distributed applications over dynamic MANETs present significant challenges in feasibility and scalability. First, tactical MANET hardware is typically experimental and available only in small quantities, so conducting evaluation of more than a few real nodes is either infeasible or impractical for cost reasons. Second, running large-scale evaluations for real tactical MANETs in a physical terrain is very costly in terms of logistics, therefore the number of tests that can be performed would less inadequate for rigorous testing.

One approach is to conduct testing over a testbed in a laboratory environment, where the testbed behaves like a real MANET to the applications. Our goal has been to develop testbed technologies that enable testing and evaluation of any software application on any common operating system (OS), given any MANET scenario.

This paper presents the testbed construction technology developed under the *Virtual Ad hoc Network (VAN)* project to test unmodified applications over MANETs, with *a special focus on supporting network-aware applications*. The VAN Testbed technologies allow multiple application instances running on virtual nodes, possibly hosted on the same physical machines or on different ones, to send IP packets to each other via a simulated MANET, as if each application instance runs on a real mobile node and its packets are transmitted over a real MANET.

Our main contributions so far have been to: a) enable *transparent packet forwarding* between Xen-based virtual hosts and OPNET-based network simulation, and b) enable highly-automated and usable *management instrumentation* of the network simulation (for individual stacks and the entire simulated network) via a standard interface (SNMP).

The transparent forwarding capability is enabled by a special kind of node emulation in which the IP protocol stack

[1] The research reported in this document/presentation was performed in connection with contract number W15P7T-08-C-P213 with the U.S. Army Communications Electronics Research and Development Engineering Center (CERDEC). 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 CERDEC, 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.

978-1-4244-5239-2/09/$26.00 ©2009 IEEE

functions are split between Xen-based virtual hosts and simulated virtual nodes, with the split point within the IP layer. Further, we have used Layer 2 packet forwarding with MAC address translation to provide more flexibility and increased performance.

The management instrumentation capability consists of automatically generated SNMP agents that are deployed on each virtual host for managing that part of the split stack implemented in the simulation, thus enabling network management applications to execute over a VAN testbed as in a real MANET. Additionally, a Wizard Management agent provides access to the entire simulated network for software testing activities.

The remainder of the paper is organized as follows: Section 2 discusses related work. Section 3 describes the problem background and relevant technologies. Section 4 presents the architecture and functionality of a VAN testbed, with the associated concepts and objectives. Section 5 briefly discusses *functional* evaluation of a VAN testbed in production testing and overhead data. Section 6 presents the ongoing work and future plans, and Section 7 concludes the paper with a summary.

2. RELATED WORK

Prior work applicable to functional testing and performance evaluation of distributed software systems can be roughly divided into the following categories by the characteristics of the approaches.

Specialized network testbeds: This type of approaches addresses testing a network consisting of some specific hardware and/or software components (e.g., radios and/or OSs). Some approaches allow the use of real distributed systems for testing (e.g., systems that rely on the combination of TinyOS and Mote sensors [8], while others only allow models for a specific network simulator to be used as test targets (e.g., ns-2 models over a miniaturized 802.11 radio testbed [9]). The VAN Testbed approach aims to be *as generic as possible* with respect to simulators, models, and OSs/applications.

Full network emulation: An emulated network, from the viewpoint of packet forwarding, introduces various conditions (i.e., loss, delay, jitter). When a network is "emulated", both the packet forwarding process and network protocol interactions are neither real nor being modeled as stepwise transitions using finite state machines reflecting actual protocols. Often, protocol stack interactions are modeled by programs offering abstracted behaviors, e.g., a packet entering the emulated network is either regarded as dropped along the route, or delivered to its destination node after certain delay, with or without error. To use an emulated network for testing of real distributed software, packets generated by real software must be "injected" into the emulated network as if routed into a real network. [10, 11, 12, 13, 14, 15] give examples of network emulation. A high level of abstraction in these approaches typically restricts their use with network-aware applications.

Hybrid network simulation: approaches split the network protocol stack simulation onto two different machines and provide a mechanism that creates a virtual network that behaves identically to a real network from the perspective of real applications (e.g., [16]). The VAN Testbed technology uses an approach of this type.

More recently, significant work has been done on complete testbed technologies that combine generic network emulation/hybrid simulation with real OS and application code, most notably in [17] and [18]. Detailed feature and performance comparison of our approach with these technologies is beyond the scope of this paper; however, neither, at the time of writing, includes support for slower-than-real time execution of unmodified applications or management instrumentation support for network-aware applications.

3. BACKGROUND

In this section, we explain the key technologies that provide the backbone of this paper and have been used in construction of VAN testbeds.

A. Network Simulation and Network Emulation

Two of the most commonly used experimental techniques for the design and validation of new and existing networking technologies are network simulation and network emulation. In network simulation, a program [14] models the behavior of a network by reproducing the interactions between the different entities (hosts/routers, data links, packets, etc) using mathematical formulas, and replicating behavior of real network components with desired fidelity. Network emulation [6, 7] combines real elements of a deployed network, namely, the end hosts and protocol implementations, with highly abstracted network representations of network links, intermediate nodes, and background traffic. The fundamental difference between simulation and emulation is that while the former runs in virtual simulated time, the latter must run in real time. A simulated network can offer higher fidelity than an emulated one. VAN testbeds employ simulation-based virtual networks.

B. Software-in-the-loop Simulation

Software-in-the-loop (SITL) simulation is a methodology that utilizes network simulation models to evaluate the effectiveness and scalability of real software as it is deployed in the field [13]. This results in high fidelity experiments as there is no implementation-related inconsistency between the deployed and the in-the-loop software, which also happens to get tested more extensively in a close-to-

real simulated environment. VAN testbeds use network simulators for modeling MANETs and the software-in-the-loop consists of the applications under test.

C. Network Simulator and Models

Currently the VAN Testbed technology supports network models developed with [3], while support for [4] is under development. The technology is model-neutral by design and supports execution of third-party models with little or no manual conditioning. To be executed in a VAN testbed, a network model should implement the lower part of the IP stack, including a mobile link layer and possibly a mobile network layer responsible for end-to-end packet forwarding. So far we have experimented with models including different TDMA/OFDMA mobile link layers and ad hoc routing protocols, with unicast and broadcast traffic.

D. Virtualization

In computing, virtualization is a broad term that refers to the abstraction of computer resources. Virtualization makes it possible to run multiple OSs on the same computer at the same time. VAN testbeds use Xen-based host virtualization for executing multiple virtual hosts on a single physical machine. Xen [5] is an open-source hardware/paravirtualization method which provides an abstraction layer that allows a physical host to execute one or more virtual hosts, effectively decoupling the OS and its applications from the underlying physical machine. Xen currently supports a wide range of guest OSs (Linux, Solaris, BSD variants), and Windows (hardware mode) on several CPU architectures.

4. A VAN TESTBED

The VAN Testbed technology is designed to allow the testing and evaluation of applications over MANETs. A VAN testbed places emphasis on: a) support for executing unmodified applications, i.e., without requiring code change to accommodate the testbed; b) fidelity–providing an accurate representation of the network that is virtualized; and c) testbed scalability–enabling a large number of copies of the same application to execute over the testbed. In order to support testing of unmodified applications, a VAN testbed provides an environment context that is as close as possible to the real deployment environment. Accordingly, in a VAN testbed each application instance executes over its own OS instance, thus having its own set of environment variables, libraries, configurations, and file system(s). This separation is designed to model a real deployment environment where each instance in the testbed represents a real mobile. In order to provide fidelity, a VAN testbed features an emulated network consisting of a network simulation model that executes in a network simulator in real-time (execution of a network model slower than real time is an ongoing effort). The network simulator employs a SITL technique to convert the packets generated by the applications into simulated packets to pass through the simulated network and again to real packets.

The use of a simulated network provides several advantages over a more abstract network emulator. First, simulators can offer highest fidelity by simulating the detail of packet forwarding process.

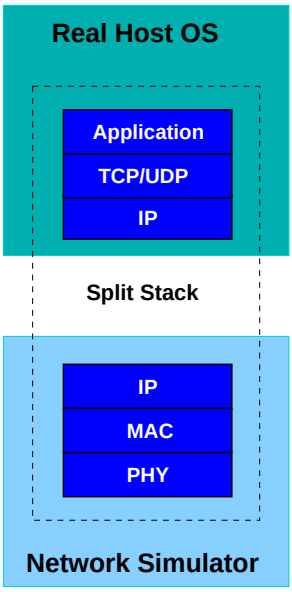

Figure 1: The concept of a split stack

Secondly, major network simulators usually have a multitude of simulation models built for prior simulation studies, thus allowing a VAN testbed to re-use such models and scenarios, with the desired degree of fidelity. Thirdly, when simulating a network, a simulator employs network scenarios containing a collection of all simulated nodes interacting with each other, where each simulated node implements the functionality corresponding to a real ad hoc node. The modeling of all network nodes provides an opportunity to test *network-aware* applications that exercise monitoring and control functions on the nodes (e.g., network management applications). By contrast, network emulators that model an entire network as a black box between endpoint nodes cannot support such testing.

A. Transparent Forwarding

In order to allow unmodified applications to exchange packets over a simulated network, a VAN testbed introduces the concept of a *split* network stack. In a split stack (Figure 1), the stack is split into two parts, where the higher layers are implemented by the OS that hosts applications, while the lower layers are implemented in the network simulator. In our implementation, the stack is split in the IP layer: the transport layer and packet encapsulation functions of the IP layer are provided by the host OS, while the simulator implements the rest of the stack.

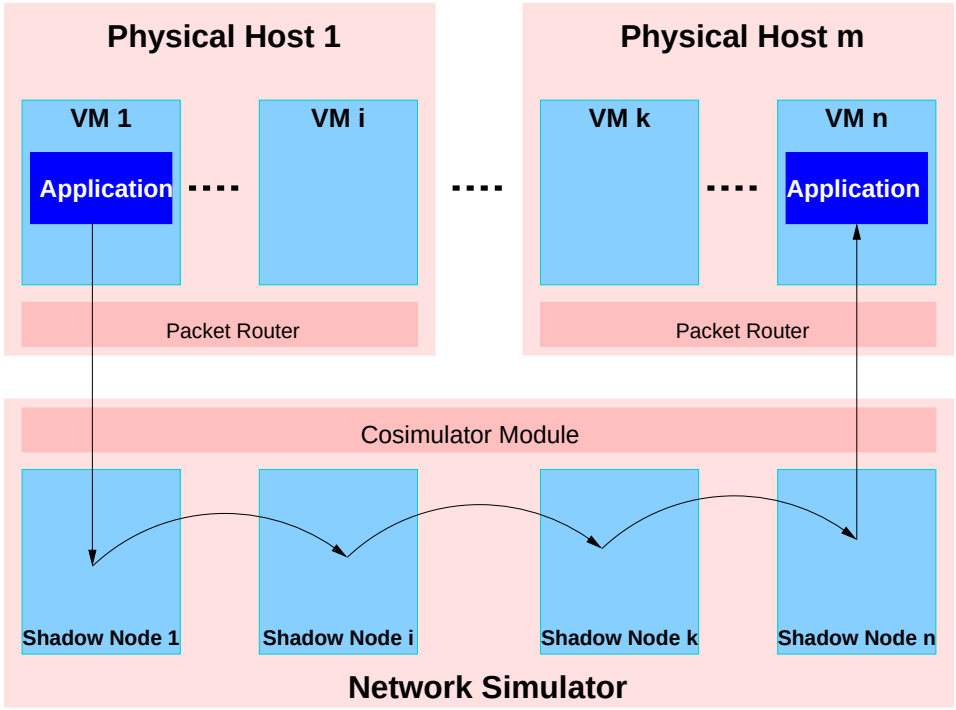

Figure 2: Transparent forwarding in a VAN Testbed

The simulator's IP layer consists only of IP encapsulation and enables creation of simulated IP packets out of real IP packets. This configuration sets up an architecture for forwarding application-generated IP packets between the upper stack and the lower stack, as well as for accessing lower stack parameters of each simulated node for network-aware applications. While the split stack concept alone allows the execution of unmodified applications over a simulated network, large-scale scenarios would require a prohibitively expensive testbed employing a large number of corresponding physical hosts if one physical host were used to represent one host in a scenario.

To address this issue, a VAN testbed uses host virtualization so that multiple VMs can execute on the limited hardware resources. This technique allows each instance of the application to execute in a separate and distinct environment, nearly identical to that provided by a real host at deployment. The upper half of the split stack is implemented in the virtual machines.

Figure 2 presents transparent IP packet forwarding in a VAN testbed. Multiple VMs, VMi, are installed on different physical hosts. Figure 2 presents transparent IP packet forwarding in a VAN testbed. Multiple VMs, VMi, are installed on different physical hosts.

The network simulator employs a network scenario consisting of shadow nodes, each implementing the lower layers of the stack for its corresponding VM. Packets generated by an application are transferred to the simulator and injected in the corresponding shadow node, and transformed into simulated packets. Subsequently the simulated packets are forwarded to their destination, according to the simulation scenario. Finally, the packets are transformed into real packets and transferred to the destination VM and its corresponding application instance. A testbed employs a Packet Router module on each physical host and a Cosimulator module loaded within the network simulator process. The Packet Router is responsible for transferring the IP packets between the virtual machines and the simulator process with a low overhead. The Cosimulator is responsible for transformation between real packets and simulated packets, as well as control of simulation speed and synchronization with real time.

B. Local Stack Management

One of our main objectives is to enable applications to execute over VAN testbeds in a highly realistic environment, with no modifications to either OS configuration or application code/configuration. Accordingly, network-aware applications under test should be able to access and control the network through the same interfaces as used in their real environment.

In order to support such capability, VAN testbeds currently provide access to lower stack parameters using Simple Network Management Protocol (SNMP) (other interfaces, e.g., *sysctl*, are a subject of ongoing work and future plans). SNMP is the de-facto standard for network management and has been our first target. However, the technology we use for accessing lower stack parameters easily generalizes to other interfaces.

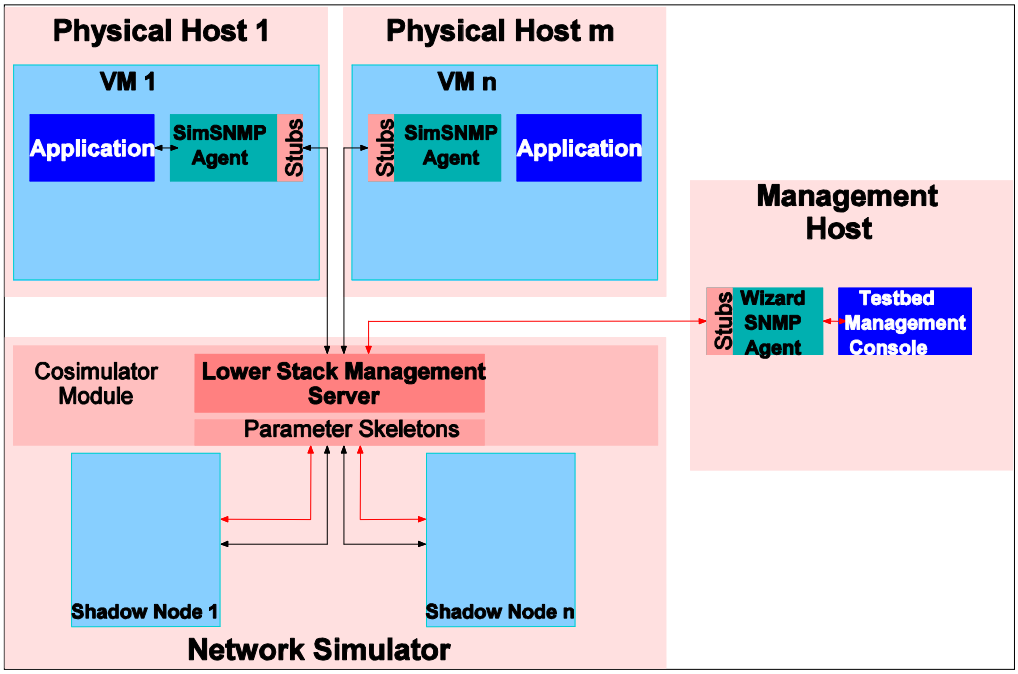

Figure 3: Access to simulated stack parameters

A VAN testbed provides an application running on a VM with standard SNMP interfaces for accessing lower stack parameters implemented in the VM's shadow node. Figure 3 shows the interaction between an application and the simulated network, and the pertaining testbed modules.

Every VM hosts a special SNMP agent, SimSNMP, that exposes standard interfaces for accessing the instrumented parameters. Every SNMP request for a lower stack parameter is transferred to the simulator process where it is handled and the results are returned to the SimSNMP agent and subsequently to the application.

SimSNMP is implemented using [6], and the communication between the agent and the simulation process is carried out via CORBA, facilitating automatic adaptation for different parameters available in network models.

The VAN Testbed technology has been designed to be as general as possible, supporting various network models. More precisely, a testbed should be model-neutral, i.e. support the execution of third-party network models with little or no manual conditioning. While the parameters provided to the applications, their data types, and actual access implementation depend on the actual model, the infrastructure for providing SNMP access for unmodified network-applications is a highly automated process, in which the SimSNMP agent is constructed from an SNMP management information base (MIB) specification and linked with an instrumented simulator model. No knowledge of SNMP beyond MIB definition is assumed, so the effort to test a new application over a new, possibly third party network model is minimized.

C. Network Scenario Monitoring and Control

Application testing over a MANET requires the ability to monitor and, even more importantly, to create at will the network conditions that exercise application features and error handling. An ability to do this in a scripted mode is provided by the simulation scenarios. However, a tester often needs an ability to introduce conditions interactively (e.g., failures, information assurance events) over the course of a long-running test scenario. To enable this ability, we provide tools for building and execution of a Wizard Management (WM) agent, a module designed for monitoring and steering the entire simulated network at run time.

Figure 3 shows a WM Agent executing on a Management Host, where an operator oversees the execution of testing scenario. A WM agent is similar in construction to a SimSNMP agent, and is also automatically built from an SNMP MIB. A WM agent, however, provides access to all the nodes in the simulated network by exposing the node identifier in its MIB. Moreover, the set of parameters available to the WM agent is usually a superset of those provided via SimSNMP agents. Such extra parameters could be those: a) not useful to network-aware applications, b) not reflected in the real network but representing a byproduct of modeling the network, or c) used for injecting events (controlled errors, faults) in the simulated network. A WM agent can also be used in a scripted mode via command-line SNMP utilities.

D. Test Scenario Monitoring and Control

In a test scenario, a network scenario is combined in the same timeline with an application execution scenario. In a fully interactive mode, the tester needs an ability to communicate with applications under test using the application's UI. This is impractical in large scale testing, where this needs to be scriptable, and the set of scripts constitutes an application execution scenario in the fully scripted mode. A mixed mode where most of the application's execution is scripted, but some is interactive is also useful.

To enable the basic capability of the tester to interact with applications in a VAN testbed, each VM is given a second network interface which is not subject to transparent forwarding, a *management network interface* (the same network interface is currently used for SimSNMP communication). The management network interfaces are configured to place VMs on the same LAN as one or more management hosts. The tester is then able to use tools installed under the guest OSs in the VMs and the management host's OS (e.g., remote shell/desktop) to interact with the applications in any mode. Simple tools to deploy and execute scenario scripts on all or selected VMs are also provided; a sophisticated UI is the subject of future work.

E. Testbed Resource Management

The basic capabilities to start, suspend/resume, stop, and monitor VM execution are provided by Xen. Currently, these are enhanced with a simple centralized scripting capability over *ssh*. However, software configuration of a VMs for a large-scale testing scenario can be quite complex and differ widely between scenarios. Enabling scalable and repeatable VM configuration ultimately requires a highly automated capability to bring any VM to a desired configuratuion state, including OS kernel and userland, installed support libraries, started services, network and service configuration, and, finally, the configuration of the software under test itself. A significant part of out ongoing and future work is focused in this area.

5. EVALUATION

In order to evaluate the functionality and viability of the VAN testbed for testing tactical applications, we have provided a testbed for production software testing of a number of different tactical applications being developed at Telcordia. These applications (briefly described next) have been tested extensively by *its testers* on a variety of mobility scenarios for MANETs of 10 to 50 nodes, proved functional usefulness of a VAN testbed in production testing and allowed us to formulate future requirements.

DRAMA [10] is a policy-based MANET management system, composed of autonomous agents executing on, and managing each mobile node according to a set of policies. DRAMA agents organize themselves into a management hierarchy that helps disseminate policies through the network and propagate management information.

IFC (Integrated Fault Correlator) is a network management application designed to detect and diagnose faults in the network. IFC instances execute on each mobile node, polling local node configuration/status, and intercommunicate to infer the status of the entire MANET.

MPT (Mission To Policy Translator) is a network management application designed to configure/reconfigure the network according to a mission plans and taking into account actual conditions in the network. MPT also monitors and configures each node in the network.

QoS Manager is an FCS component that performs dynamic, measurement-based admission control in the application layer. It is a network-*un*aware application in the sense of local stack management.

In addition to evaluating the functionality of the VAN testbed as a test tool, we have measured the overhead introduced by the testbed under several benchmarks (ping and TCP-based video). Under moderate traffic conditions of around 50 Mbps (aggregate), the VM-to-VM packet latency introduced by the testbed ranged in the 130-220 microseconds. Under more aggressive traffic conditions, the latency was 330 and 510 microseconds for loads of 80Mbs and 120 Mbps respectively.

6. ONGOING AND FUTURE WORK

The VAN Testbed technology is under active development. In *transparent forwarding*, we are moving towards a more scalable and automatable approach based on VLAN tagging and physical routers that will greatly simplify Xen physical host configuration and reduce their load. We are also working on support for multicast and extending our implementation to support [19] in the same fully transparent manner as already achieved for [3].

To achieve the desired level of scalability in the number of nodes while supporting high fidelity simulation models, we are working on a Xen-based VM-to-simulation time synchronization mechanism that will allow a VAN testbed model to execute slower than real time. We have previously implemented a similar OS-specific capability for the DRAMA system and [3] and showed that networks of up to 500 nodes can be supported with a single Modeler instance in slower-than-real time. The ongoing work will make this OS-independent for the Xen-supported guest OS set and extend this capability to network models for [6].

A side effect of the time synchronization capability will be the ability of the tester to pause a running test scenario at will or define breakpoint(s), inspect the state of the running network and/or application execution scenario or op-

tionally change state via the Wizard Management agent, and resume execution. These are standard capabilities of software debuggers, and we plan to provide them to the developers of the distributed tactical applications.

In *local stack management* and *network scenario monitoring and control*, we are enhancing the tools for automated management agent generation to support [6] and plan to support the *sysctl* management interface under Linux and BSD guest OSs.

Finally, in *testbed resource management*, we are working on a number of techniques to allow rapid, centralized configuration of an entire testbed for a given test scenario, including the use of disk farms, file systems shared by multiple VMs, and checkpointing configuration states.

7. CONCLUSIONS

In this paper we describe the research and development progress on building a virtual ad hoc network testbed. The testbed supports testing unmodified applications running in their own native OSs, while the communication packets are transmitted through a simulated mobile ad hoc network. We consider this a critical technology for testing and evaluating functional correctness and communication performance for applications over the future tactical networks. The follow-up research is to focus on enabling the testbed for testing unmodified applications at a large-scale (500-1000 instances), where the differences between the simulation time and wall clock time must be addressed.

ACKNOWLEDGEMENT

We would like to sincerely thank the Office of the Secretary of Defense for sponsoring this work.

REFERENCES

[1] P. K. Biswas, C. Serban, A. Poylisher, J. Lee, S. Mau, R. Chadha, and C. J. Chiang, "An Integrated testbed for Virtual Ad Hoc Networks," in Proc. TRIDENTCOM 2009.

[2] J. Case, R. Mundy, D. Partain, and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework," RFC 3410, Dec. 2002.

[3] OPNET, "Modeler," 2009. [Online]. Available: http://www.opnet.com/solutions/network rd/modeler.html

[4] Scalable Network Technologies, "QualNet," 2009. [Online]. Available: http://www.scalable-networks.com/products/developer.php

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in Proc. SOSP '03, 2003.

[6] Net-SNMP Team, "Net-SNMP toolkit," 2009. [Online]. Available: http://www.net-snmp.org/

[7] "Policy-based mobile ad hoc network management," in Proc. of IEEE POLICY '04. Washington, DC, USA: IEEE Computer Society, 2004, p. 35.

[8] D. Johnson, T. Stack, R. Fish, D. M. Flickingery, L. Stoller, R. Ricci, and J. Lepreau, "Mobile Emulab: A Robotic Wireless and Sensor Network Testbed." in Proc. IEEE INFOCOM'06.

[9] P. De, A. Raniwala, S. Sharma, and T.-C. Chiueh, "MiNT: A Miniaturized Network Testbed for Mobile Wireless Research," in Proc. IEEE INFOCOM' 05.

[10] M. Carson and D. Santay, "NIST Net: a Linux-based network emulation tool," ACM SIGCOMM Computer Communication Review, vol. 33:3, pp. 111–126, 2003.

[11] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker, "Scalability and accuracy in a large-scale network emulator," in Proc. USENIX OSDI'02.

[12] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in Proc. USENIX OSDI'02.

[13] Y. Zhang and W. Li, "An integrated environment for testing mobile ad-hoc networks," in Proc. ACM MOBIHOC'02.

[14] P. Zheng and L. M. Ni, "Empower: A network emulator for wireline and wireless networks," in Proc. IEEE INFOCOM'03.

[15] J. Zhou, Z. Ji, and R. Bagrodia, "TWINE: A Hybrid Emulation Testbed for Wireless Networks and Applicagtions," in Proc. IEEE INFOCOM'06.

[16] J. Zhou, Z. Ji, M. Takai, and R. Bagrodia, "MAYA: Integrating hybrid network modeling to the physical world," ACM Transactions on Modeling and Computer Simulation, vol. 14(2), 2004.

[17] Emulab Team, "Emulab: total network testbed," Flux Group, School of Computing, University of Utah, 2009. [Online]. Available: http://www.emulab.net

[18] Network and Communication Systems Branch, "MANE publications," Naval Research Laboratory, 2008. [Online]. Available: http://cs.itd.nrl.navy.mil/pubs/

[19] Scalable Network Technologies, "EXata," 2009. [Online]. Available: http://www.scalable-networks.com/products/exata.php