Realistic Modeling of Tactical Networks with Multi-Level Security in VAN Testbeds

Alex Poylisher, Taichuan Lu, Constantin Serban, John Lee, Ritu Chadha, Cho-Yu Jason Chiang
Applied Research, Telcordia Technologies
Piscataway, NJ, USA

Kimberly Jakubowski, Keith Whittaker, Rocio Bauer
S&TCD, U.S. Army CERDEC
Fort Monmouth, NJ, USA

***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 Virtual Ad hoc Network (VAN) testbeds[1] that can imitate a tactical MANET to a high level of fidelity. For applications, this is achieved by executing unmodified software in their native operating systems on the machines over an emulated platform that uses identical logical layout of the real physical platform; 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 analyze the challenges and tradeoffs of realistic modeling of tactical networks with multilevel security (MLS), and present a case study of modeling BCTM-like manned ground vehicle (MGV) platforms in the VAN Testbed. Lessons learned during application development and testing performed on the model are also discussed.**

Keywords-Multi-Level Security, Red-Black Network Separation, MANET, Testbed.

I. 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. Secondly, 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 application on any common operating system, given arbitrary MANET scenarios.

This paper describes an extension of our previous work on Virtual Ad hoc Network (VAN) testbed technologies for testing unmodified applications over MANETs; [1-6] provide the background, including comparison with other testbed approaches. The focus of the paper is on enabling support of modeling multi-level security enclaves on mobile platforms equipped with one or multiple radios. The purpose is to allow applications that are supposed to run in different security enclaves to send IP packets in a lab testbed environment. The application communications are subject to the security constraints as if they were deployed on real platforms each of which hosts multiple security domains. For example, applications in a lower-security domain will not be allowed to communicate with applications in a higher-security domain without exception rules pre-installed in the cross-domain guard. As another example, applications on the red-side of a red-black separated network have very limited access to the information on the black side, if it is permitted.

Our main contributions on augmenting the VAN testbed to support multi-level security are as follows: i) we developed an architectural framework that enables the modeling of communications in a Multi-Level Security (MLS) environment with adequate fidelity without relying on restricted-use software, ii) we streamlined the process of such modeling by automating the grouping of multiple virtual machines that represent BCTM-like MGV platform hosts and mapping them to the radios on the platform in the simulated network.

The remainder of the paper is organized as follows: Section II gives a brief overview of the architecture and functionality of a VAN testbed; Section III discusses the challenges of modeling multilevel security in tactical networks; Section IV presents our current approach of modeling a BCTM-like MLS network in a VAN testbed; Section V discusses how network monitoring is performed in the testbed and the simulated network that was used to support the study. We conclude the paper with a discussion of possible future work and a summary.

[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.

II. Overview of the VAN Testbed

A. Architecture

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 in a Xen [11] virtual machine (VM) over its own OS instance, having its own set of environment variables, libraries, configurations, and file system(s).

In order to ensure high network fidelity, a VAN testbed features an emulated network consisting of a network simulation model that executes in a network simulator ([9] and [10,13] are supported) in real-time. The simulator employs a SITL technique to convert the packets generated by the applications into simulated packets to pass through the simulated network, and again converts them back to real packets when they exit the simulated network. 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 as packets travel through the protocol stacks and from one node to another. 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 of 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 do not allow such testing. The use of a simulated network provides several advantages over a more abstract network emulator.

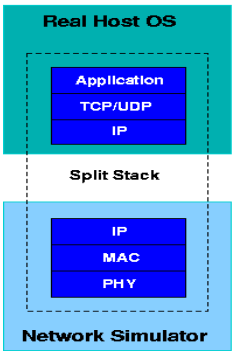

Figure 1. The split-protocol stack approach underlying the VAN testbed

B. Transparent Forwarding and Host Virtualization

In order to allow unmodified applications to exchange packets over a simulated network, a VAN testbed introduces the concept of a split protocol stack. A split protocol stack, shown in Figure 1, is split into two parts where the higher layers of the stack are implemented by the OS that hosts applications while the lower layers of the stack are implemented in the network simulator.

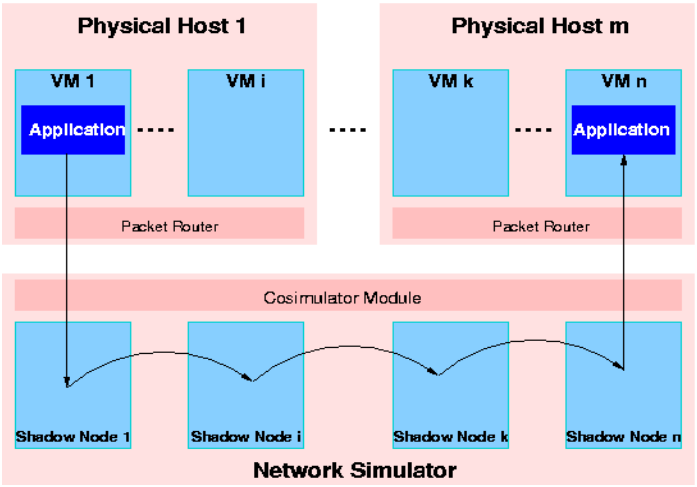

Figure 2. Transparent packet forwarding in the VAN testbed

In the current VAN implementation, the stack is split at the IP layer: the transport layer and packet encapsulation function of the IP layer are provided by the host operating system, while the simulator implements the rest of the protocol stack. The simulator's IP thus 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 testing and evaluation scenarios would require a prohibitively expensive testbed employing a large number of corresponding physical hosts. In order to address this issue, a VAN testbed uses host virtualization so that multiple virtual hosts 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 illustrates the concept of transparent IP packet forwarding in a VAN testbed. Multiple virtual machines, 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 a corresponding VM. Packets generated by an application are transferred to the simulator, injected in the corresponding shadow node, and transformed into simulated packets. Subsequently the simulated packets are forwarded to their destination,

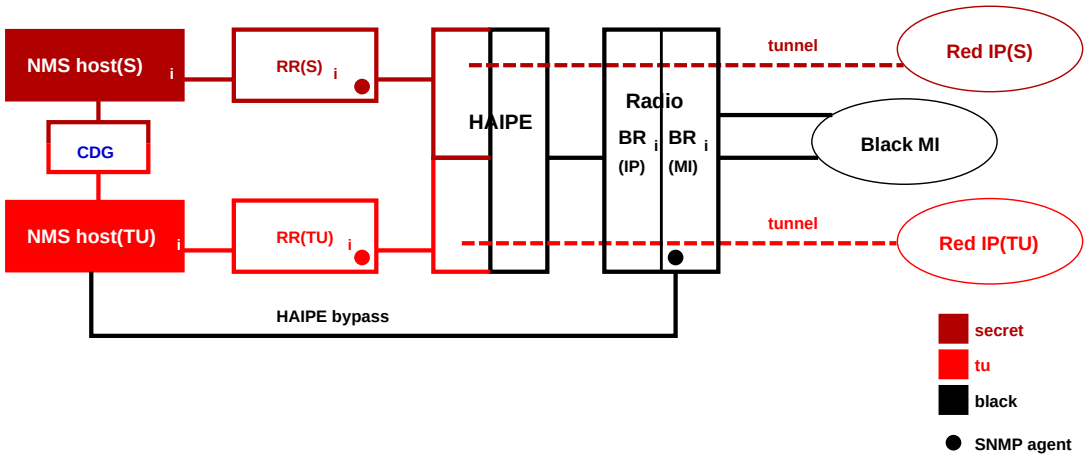

Figure 3 BCTM-like reference MGV platform layout with red-black separation.

according to the simulation scenario.

Finally, the packets are transformed into real packets and transferred to the destination VM and its correponding application instance. The testbed employs a Packet Router module on each physical host and a Cosimulator Module loaded within the same process as the network simulator. The Packet Router is responsible for transferring IP packets between the virtual machines and the simulator process with a low overhead. The Cosimulator Module is responsible for, among other things, the transformation between real packets and simulated packets, as well as control of simulation speed and synchronization with real time.

III. MODELING MLS IN VAN TESTBED

A. Reference MGV Platform Layout

Figure 3 shows a BCTM-like reference platform layout with red-black separation adopted for the current evaluation. The layout includes 2 red enclaves (TU and Secret), each represented as a single NMS host for simplicity, a Cross-Domain Gateway (CDG) between the two enclaves, two red routers (RRs), the High Assurance Internet Protocol Encryptor (HAIPE) device (logically two devices, one per red enclave), the black radio (logically split into the IP and Mobile Internet (MI) layer black routers (BRs). The figure also shows the black MI network and the HAIPE tunnels established over the black IP network to other platforms, for each of the two enclaves.

Additionally, a HAIPE bypass interface connected to the MI black router allows read-only access to certain management information in the MI black router from the TU enclave, and a management agent is running in a red router. For simplicity and without loss of generality, the reference platform layout assumes that access to management information is available via SNMP [8]. The SNMP agents are shown with dots in the RRs and the MI BR in the Figure. The two red enclaves can exchange only a strictly controlled and predefined set of messages via the CDG.

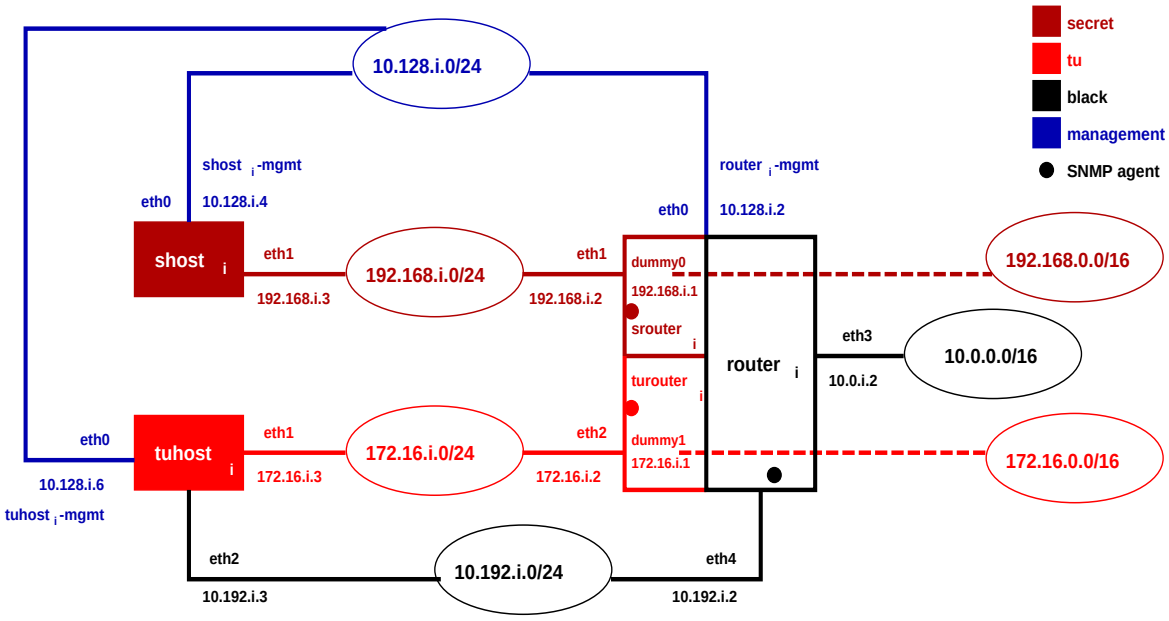

Figure 4: The model of the reference platform used in the VAN testbed.

B. Experiment Setup

Figure 4 shows the model of the reference platform layout as implemented in the testbed. The index i denotes to the platform identifier. The Figure shows the two enclaves as shost and -tuhost virtual machines (VMs), two in-platform private networks (192.168.i.0/24 for the Secret enclave, 172.16.i.0/24 for the TU enclave), and the bypass private network between the tuhost and the MI BR (10.192.i.0/24).

The HAIPE device and IP BR have been modeled in a single VM as follows. For each red enclave, an additional software-only network interface has been defined in the VM (dummy0 and dummy1 for the Secret and TU enclaves respectively). These two interfaces have been used as IPSec tunnel endpoints (Linux racoon) for a full mesh of tunnels defined from the Secret enclave in a given platform to the Secret enclaves in all other platforms, and similarly from the TU enclave in a given platform to the TU enclaves in all other platforms, both over the black IP network (10.0.0.0/16).

The tunnels are pre-configured with the lifetime of a typical FCS mission (72 hours) and pre-shared keys and are established on platform startup. As HAIPE is based on IPSec with additional cryptography features, IPSec provides an adequate level of fidelity for the current evaluation in terms of rough latency, some encryption overhead, and possible failures. IPSec parameters were set up to accommodate the relatively high loss and latencies typical of the WNW networks, as follows, and to avoid tunnel teardown (which would not occur with real HAIPE devices):

- Max number of retries: 30
- Max interval to re-send: 10 sec
- The number of packets per send: 1
- Timers for phase completion: 120 sec for both Phase 1 and Phase 2.
- NAT keep-alive: off.

The two RRs have been modeled as static routes between eth1 and dummy0 for the Secret enclave, and eth2 and dummy1 for the TU domain. The SNMP agents in the RRs allow introduction of failures in the RRs by failing these interfaces or routing between them, or both.

C. Support for network-aware applications

One of the goals of multi-level security modeling in the VAN Testbed is to provide applications with the illusion of operating in a real, security enabled platform environment. Applications operating in the VAN Testbed will thus be exposed to the same interfaces available in real platform deployments. While the addressing scheme presented above ensures a high fidelity platform-to-platform and enclave-to-enclave communication, many applications require explicit interaction with specific elements in the network. For example, network configuration and network monitoring applications interact with existing devices on the platform to enable certain functionality or diagnose the status of the network. To this end, we have instrumented each of the Red Routers and the Black Router to provide the expected management interfaces for administrative purposes.

The management interfaces are implemented as SNMP agents that provide access to the administrative and operational status of the routers, as well as per interface basis. The functionality of the router and its management operations is implemented using *iptables*.

Another goal of our detailed multi-level security modeling of a BCTM-like platform is to evaluate the behavior of the application software, as well as of the network in general under faulty conditions. While the VAN Testbed previously provided high fidelity modeling of various failures of the black radio, failures occurring within the platform were ignored. In this work we augmented the failure model to support the injection of failures on any Red Router (Secret and TU), as well as global or per interface failure of the Black Router. We used the same mechanism for injecting and clearing faults as in the case of network management operations: specialized MIBs, called Wizard MIBs, were provided for the SNMP agents in each router for inducing/clearing a failed state. Again, implementation of a failure has been emulated by preventing traffic with *iptables*.

In order to facilitate the activities related to the setup/initialization of a scenario involving a desired set of platforms, the following activities are scripted:

- creating/destroying emulated hosts within platforms
- creating/assigning interfaces within the emulated hosts
- creating/refreshing IPSec tunnels for each enclave
- starting SNMP agents for each emulated router device (Red Secret, Red TU, and Black.)
- setting up iptable rules for marking traffic class for specific applications.

In addition to the above scripts that create a complete scenario consisting of emulated platforms, we have created scripts that allow for a convenient way of injecting faults in any of the platform elements described above, as well as on the black network. The scripts rely on the management interface (10.128.i.0/24) available on each emulated host, which allows access to the wizard MIB of each device from a centralized location.

IV. Monitoring the Simulated Network

One of the main objectives of VAN is to enable applications to execute over the VAN testbed in the same way as they would if they were executing in a real MANET, with no modifications to either OS configuration or application code/configuration. Accordingly, network-aware applications under test should be able to access, monitor, and control the network through the same interfaces as used in their real environment. In order to support such capability, the VAN testbed currently provides access to network parameters using Simple Network Management Protocol (SNMP). SNMP represents the de-facto standard for network management, and the majority of the network management applications executing on top of a VAN testbed use SNMP. However, the technology used for accessing network parameters can easily apply to other interfaces.

A VAN testbed provides every application running on a VM with standard SNMP interfaces for accessing network parameters implemented in the VM's shadow node (i.e. the node that represents a real node within the simulated network). Figure 5

shows the corresponding interaction between an application and the simulated network, and the pertaining modules.

ing the node identifier in the MIB, thus enabling the steering and monitoring of the entire network.

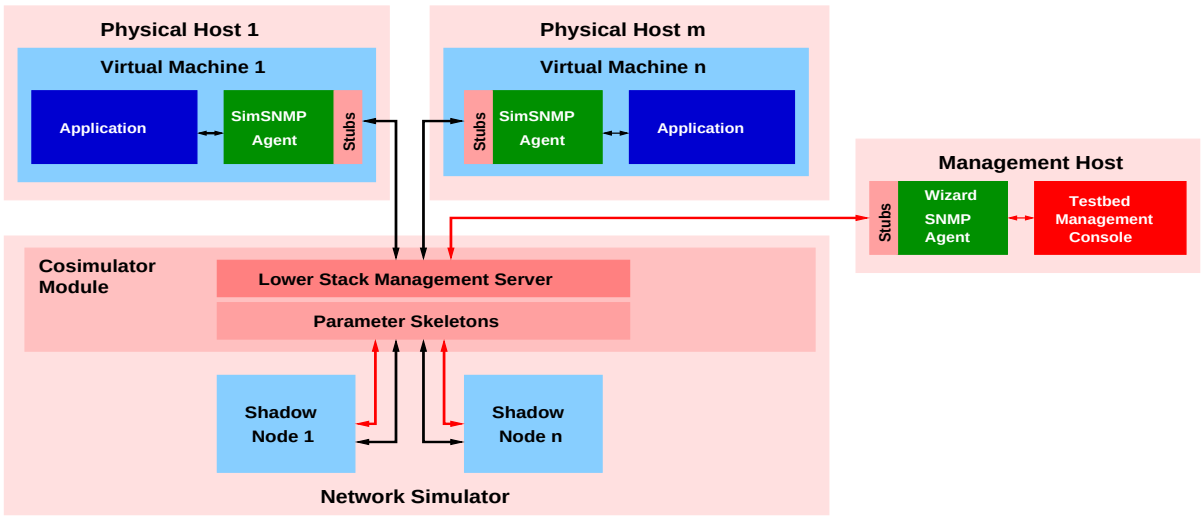

Figure 5: Access to simulated stack parameters

Every VM in the testbed hosts a special SNMP agent, SimSNMP, that exposes standard interfaces for accessing the instrumented parameters. Every SNMP request for a MIB value is transferred to the simulator process where it is handled; the results are returned to SimSNMP and then to the application.

SimSNMP is implemented using [12], a widely used open source package available for most major operating systems. The communication between SimSNMP and the simulation process is carried out through CORBA calls, facilitating automatic adaptation for different parameters available in network models.

V. Monitoring and Control

A. Network Scenario

Testing an application such as the IFC over a MANET requires the ability to monitor and, even more importantly, to create at will the network conditions that give rise to network faults and clearing of those faults. An ability to do this in a scripted mode is certainly provided by the simulation scenarios. However, a tester often needs the ability to introduce conditions interactively (e.g., failures and performance problems) over the course of a long-running test scenario. To enable this ability, the VAN testbed technology provides tools for building and executing a Wizard Management agent, a central SNMP agent designed for monitoring and steering the entire simulated network at run time.

Figure 5 shows a Wizard Management Agent executing on a Management Host, which is the host where an operator oversees the execution of testing scenarios in a VAN testbed, its applications under evaluation, and the emulated network.

A Wizard Management agent is similar in construction to a SimSNMP agent. A Wizard Management agent, however, provides access to all the nodes in the simulated network by

Moreover, the set of the parameters available to the Wizard Management agent usually represents a superset of the parameters provided through SimSNMP agents. Such extra parameters could be a) parameters not necessary to network-aware applications, b) parameters which are not reflected in the real network but represent a byproduct of modeling the network, or c) parameters used for injecting events (controlled errors, faults) in the simulated network. Of course, a Wizard Management agent can be used in a scripted mode via command-line SNMP utilities.

B. Test Scenario

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 user interface. This is impractical in large-scale testing, where the same ability 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 the tools installed under the guest OS in the VMs and the management host's OS (e.g., remote shell or desktop) to interact with the applications in any mode. Simple tools to deploy and execute scenario scripts on selected VMs are also provided.

C. Simulation Model

As mentioned earlier, the present stage of design and the limited availability of FCS radios, waveforms, and networks precludes the ability to demonstrate the technology readiness level of the IFC software under field conditions. Instead, we utilize a high-fidelity modeling and simulation environment. The primary network model chosen for this is the Wireless Transport Model developed by Avaliant, LLC for the FCS SSEI IPT and used in the FCS TRL-6 experiments for the FCS QoS [7]. The Wireless Transport Model (WTN) was used during the QoS TRL-6 experiments for initial characterization and transformations of offered loads, and to cross-check performance results with other models, namely SNT's CES (Communications Effects Server), for consistency during experimental dry runs and runs for record.

Suitability of the M&S environment for the FCS QoS TRL-6 experiments was assessed in both internal and external Verification and Validation and confirmed by PM FCS Modeling and Simulation Office in a formal Accreditation review [7]. The WTN models employ higher-level abstractions of routing and network self-discovery behaviors than those used by CES. WTN does not directly model the effects of network self-discovery overhead traffic, and did not employ specific terrain profiles for this demo. However, the general consistency of results observed in these two M&S environments lent confidence to the TRL-6 assessment for QoS in FCS. We believe that the same arguments are applicable to the suitability of WTN models for our TRL-6 experiments for IFC.

VI. Results, Issues, and Challenges

While the approach we took proved to be extremely useful in the functional and performance testing of the Integrated Fault Correlator, we have encountered several issues. First, IPSec tunnel establishment in a full mesh requires full connectivity prior to the start of a testing scenario. This has been addressed with a dummy fully connected simulation model that is substituted for the real simulation model prior to running the scenario.

Secondly, the relatively high loss and latencies typical of the WNW network are not accommodated well by the stock Linux IPSec parameters. This is addressed by increasing timeout values and the number of retries as described in Section III.B. Thirdly, tunnel teardown during a testing scenario is avoided by re-establishing tunnels at the start of a scenario and setting association lifetime in excess of the modeled mission time.

Fourthly, the approach has been so far tested on fairly small networks (two full meshes of 16 platforms). While we expect the approach to be scalable to a few hundreds of platforms, given the CPU and memory requirements seen so far, a more efficient implementation may be needed beyond that scale.

Finally, while static IPSec tunnels have been an adequate model for HAIPE devices in the evaluations conducted so far, a higher fidelity model that supports dynamic re-keying is desirable for some applications.

VII. Conclusions

In this paper we describe an extension of Virtual Ad hoc Network (VAN) testbed technologies for testing unmodified applications over MANETs. The focus is on enabling support of modeling MLS enclaves on mobile platforms equipped with one or multiple radios. Applications that are supposed to run in different security enclaves can exchange IP packets in a testbed, where all communications are subject to the same security constraints as if they were deployed on real platforms with the red-black network separation paradigm. We presented our approach, experiment setup, as well as functional results, issues and challenges. We have successfully used the VAN testbed extension to test a network management application with instances running in multiple security domains on a mobile platform.

Acknowledgment

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, C. J. Chiang, "An Integrated testbed for Virtual Ad Hoc Networks," in Proc. TRIDENTCOM 2009.

[2] A. Poylisher, C. Serban, J. Lee, T. C. Lu, R. Chadha, C. Y. J. Chiang. "Virtual Ad HocNetwork Testbeds For High Fidelity Testing Of Tactical Network Applications," in Proc. IEEE MILCOM 2009.

[3] A. Poylisher, C. Serban, J. Lee, T. C. Lu, R. Chadha, C. Y. J. Chiang. *A virtual ad hoc network testbed*. Int'l Journal of Communication Networks and Distributed Systems, Vol. X, 2010.

[4] C. Serban, A. Poylisher, C. Y. J. Chiang, "Virtual Ad hoc Network Testbeds for Network-aware Applications," in Proc. IEEE/IFIP NOMS 2010.

[5] C. H. Jong, C. Y. J. Chiang, T. Lu, A. Poylisher, and C. Serban. "Storage Deduplication for Virtual Ad Hoc Network Testbed By File-Level Block Sharing", in Proc. Workshop on Virtualization Technologies in Distributed Computing, 2010.

[6]. P. Biswas, A. Poylisher, R. Chadha, and A. Ghosh, "Hybrid testbeds for QoS management in opaque MANETS," in Proc. of the 4th Annual Int'l Conference on Wireless Internet, 2008.

[7] FCS SDD Plan for Verification and Validation of M&S Relevant Environment in FCS QoS Demonstration. Document D786-12538-1, Rev New, Technology IPT / SDSI IPT, 2007.

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

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

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

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

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

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