Distributed StealthNet (D-SN): Creating a Live, Virtual, Constructive (LVC) Environment for Simulating Cyber-Attacks for Test and Evaluation (T&E)

Gilbert Torres, NAVAIR Systems Command
Kathy Smith, James Buscemi, GBL Systems
Sheetal Doshi, Ha Duong, Defeng Xu, Rajive Bagrodia, Scalable Network Technologies Inc
H. Kent Pickett, MITRE Corp.

ABSTRACT

The Services have become increasingly dependent on their tactical networks for mission command functions, situational awareness, and target engagements (terminal weapon guidance). While the network brings an unprecedented ability to project force by all echelons in a mission context, it also brings the increased risk of cyber-attack on the mission operation. With both this network use and vulnerability in mind, it is necessary to test new systems (and networked Systems of Systems (SoS)) in a cyber-vulnerable network context.

A new test technology, Distributed-StealthNet (D-SN), has been created by the Department of Defense Test Resource Management Center (TRMC) to support SoS testing with cyber-attacks against mission threads. D-SN is a simulation/emulation based virtual environment that can provide a representation of a full scale tactical network deployment (both Radio Frequency (RF) segments and wired networks at command posts). D-SN has models of real world cyber threats that affect live tactical systems and networks. D-SN can be integrated with live mission Command and Control (C2) hardware and then a series of cyber-attacks using these threat models can be launched against the virtual network and the live hardware to determine the SoS’s resiliency to sustain the tactical mission. This paper describes this new capability and the new technologies developed to support this capability.

I. INTRODUCTION

D-SN has been developed for use in a distributed test infrastructure environment where live systems are located at different labs across the test community and connected via a high-speed backbone network. The Joint Mission Environment Test Capability (JMETC) [1] is an example of a classified, high-speed, Wide Area Network (WAN) that links many of the DOD Test Ranges and key development facilities. It is at these Test Ranges where the military hardware (some under development, some currently deployed) resides. Through the use of the Test and Training Enabling Architecture (TENA) middleware, the military hardware elements can be logically linked exchanging tactical information (as they would in a SoS field support mission) forming a distributed test environment (see Figure 1). D-SN is thus a key capability for representing the mission network and introducing the effects of cyber-attack on mission networks and systems in this distributed testing environment.

This paper describes the key technologies developed to support D-SN that extends the StealthNet LVC framework for cyber operation test and evaluation described in [2] to operate in a distributed manner.

Section II describes the elements available in the StealthNet system to model the tactical network and the simulated cyber-attacks that can be used to test the mission’s cyber robustness of these tactical networks.

Section III describes the technology development that was necessary to represent the full tactical network environment in a distributed test infrastructure. In this case, multiple Instances of StealthNet must run at each of the T&E installations participating in the WAN based distributed test. The “distributed” Instances of StealthNet must be synchronized so that representation of the simulated tactical network state (message location, link loading (for both RF and wired portions of the network) and message arrival timing) can be seamlessly represented between StealthNet Instances.

Section IV addresses the problems in representing the attack environment. Cyber-attacks initiated in one StealthNet Instance must be seamlessly integrated into the Network Architecture Under Test (NAUT) represented by all StealthNet Instances. Furthermore, the natural intermittent latencies of the test network infrastructure connecting StealthNet Instances must have a negligible impact on the fidelity of representation of the NAUT. Section IV also describes how pipelining synchronization between StealthNet Instances helps to minimize the impact of these WAN infrastructure latencies.

Section V describes related work efforts and Section VI provides the conclusion.
II. OVERVIEW OF STEALTHNET NETWORK AND CYBER ATTACK ELEMENTS

D-SN includes a library of high fidelity models explicitly representing the tactical network architecture at the elemental level as well as cyber-attacks and their impact on a mission thread. When considering the fidelity of these elements, it should be noted that D-SN passes messages (both tactical and network control) through the simulated elements in real time. This real time operation of the model is the key to measuring the impact of many cyber-attacks (Distributed Denial of Service (DDoS) and virus propagation) and their subsequent impacts on key communication links and simulated, as well as live, computer hosts supporting the mission thread. Table 1 provides an overview of the network and computer host models available in StealthNet. While cyber vulnerabilities, in the context of StealthNet attack models, have been represented in each element, not all operating features have been added (i.e. Routers can have their hold and forward memory explicitly represented) by a plethora of DDoS messages; however their versioning (Cisco vs. Juniper) is generic. The fidelity resolution of each of these elements has also been noted in Table 1 with the following key:

- **High**: Not only adequate for response to StealthNet modeled attacks but also representative of advances in live software and hardware device technology against cyber-attack.
- **Medium**: Adequate representation of response to StealthNet cyber-attack models but review needed for upgrades in device technology.
- **Low**: Model is mathematical and represents first order response to StealthNet cyber-attack models.

The elements shown in Table 1 are used to construct the network architecture and to add cyber protective features (firewalls, antivirus software, patched operating systems on Hosts, etc.) to the tactical network architecture. The network may be tiered (through the use of firewalls and switches) and contain both RF devices and Computer Hosts. A Computer Host model may be modified to represent a simple Data Storage Device (DSD). While the DSDs will not actually hold information, they are a necessary part of the complete network architecture. They are specifically needed to represent a cyber-attack consisting of an effort to gain access to a data server through the network’s protective architecture.

Table 1. Network and Host Elements in StealthNet

<table>
<thead>
<tr>
<th>Network Element</th>
<th>Description</th>
<th>Model Fidelity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router</td>
<td>Used to connect networks</td>
<td>Medium</td>
</tr>
<tr>
<td>Switch</td>
<td>Store-and-forward device to create a 802.3 Local Area Network (LAN)</td>
<td>Medium</td>
</tr>
<tr>
<td>Ethernet Switch</td>
<td>Wireless transmission and reception device</td>
<td>High</td>
</tr>
<tr>
<td>Firewall</td>
<td>Used to filter network traffic</td>
<td>Medium</td>
</tr>
<tr>
<td>Host</td>
<td>Computer Device that Sends and Receives Network Traffic</td>
<td>High</td>
</tr>
<tr>
<td>Data Storage Device</td>
<td>• Models activity of the user relating to cyber vulnerabilities</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Represents the specific services being run on a host</td>
<td></td>
</tr>
</tbody>
</table>

Currently, one StealthNet Instance will allow the construction of a tactical network simulation of approximately 1,000 elements running in real time on a desktop server (8 computing cores of state of the art 3.4 GHz CPUs with 32GB of RAM). Research into Army mission scenarios indicates that this element count of 1,000 (representing handheld and vehicle-based radios, C4I planning computers, firewalls at different echelons, and other network elements) is representative of the key network mission related elements supporting an Army Brigade. If a larger tactical force and network is required for a cyber-exercise (i.e. a full Army Division), D-SN can be used with a LAN or WAN to connect different Instances of StealthNet. In this case, each StealthNet Instance represents a Brigade network. See Section III for logic and “rules of thumb” on the assignment of tactical network segments to StealthNet Instances.

Table 2 contains a list of the categories of attacks that can be launched in StealthNet. As with the network elements defined in Table 1, each attack category has been rated with the following keys:
- **High**: Represents major characteristics of attack vectors and vectors that explicitly interact with the StealthNet Network element models. The vectors are flexible and can be easily modified to represent a variation of the attack category.

- **Medium**: Represents major characteristics of the attack vector category. Interacts with most of the network elements in an expected manner. Can be reprogrammed for new behaviors.

- **Low**: Represents at least one major characteristic of attack vector category. Can be expanded to other characteristics with reprogramming.

Table 2. Cyber Attack Models Representing Categories of Attack Software

<table>
<thead>
<tr>
<th>Attack Model</th>
<th>Description</th>
<th>Model Fidelity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jammer</td>
<td>Disrupts wireless communications</td>
<td>Low</td>
</tr>
<tr>
<td>Wormhole</td>
<td>Creates an artificially shortened path in a wireless network</td>
<td>High</td>
</tr>
<tr>
<td>Routing</td>
<td>Routers can be misconfigured by deleting valid nodes or adding superfluous threat listed nodes</td>
<td>Medium</td>
</tr>
<tr>
<td>Misconfiguration</td>
<td>Distributed Denial of Service (DDoS)</td>
<td>High</td>
</tr>
<tr>
<td>Virus</td>
<td>Malware attack that spreads based on user interaction</td>
<td>High</td>
</tr>
<tr>
<td>Worm</td>
<td>Malware attack that spreads based on network topology</td>
<td>Medium</td>
</tr>
<tr>
<td>Bot</td>
<td>Malware used to create botnet to support DDoS attacks</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Of particular interest are the virus and DDoS models in Table 2. The virus model has been structured to exploit many virus-related vulnerabilities found in MITRE Corporation’s Common Vulnerabilities and Exposures (CVE) database (which was also used for developing the Host models in StealthNet). Coupling this threat with the Firewall Architecture and Antivirus Model, a test engineer can quickly learn the vulnerabilities in different tactical architectures supporting a mission thread. It should be noted that the DDoS Model represents key aspects of a typical Botnet architecture. A test engineer can set up entry points in the StealthNet tactical network for DDoS attack messages and orchestrate the DDoS attack initiating different message types, formats, and Quality of Service (QoS) designations on the Bot-generated messages. These attack messages can be tailored to take advantage of particular cyber vulnerabilities in many tactical waveforms. For example, setting DDoS messages with high QoS values often has a double effect of impacting the attack target and consuming link bandwidth, since DDoS messages sent at high QoS priority displace valid tactical messages with lower QoS priority out of the networks’ transmission queues.

Table 3. Cyber Attack Threat Activities Represented in StealthNet

<table>
<thead>
<tr>
<th>Attack Tool Interfaces</th>
<th>Description</th>
<th>Tool Fidelity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eavesdropping</td>
<td>Listens to other wireless transmissions</td>
<td>Medium</td>
</tr>
<tr>
<td>Signal Intelligence (SIGINT)</td>
<td>Scans channels for wireless signals and analyzes signal priorities</td>
<td>Low</td>
</tr>
<tr>
<td>Port Scanning</td>
<td>Scans hosts to determine what ports are open</td>
<td>Medium</td>
</tr>
<tr>
<td>Network Scanning</td>
<td>Scans network to determine what hosts are on the network</td>
<td>High</td>
</tr>
</tbody>
</table>

StealthNet also allows live users to perform scanning activities to gain information about the simulated network architecture as a prelude to initiating cyber and kinetic attacks. Table 3 lists these activities with the same fidelity definitions as Table 2. When using the StealthNet system as part of a closed (information on network architecture not given to live attackers) LVC environment, Red teams can “discover” the network architecture by invoking these activities. Of particular interest are the interfaces for Port Scanning and Network Scanning. While Port Scanning only considers TCP ports, users of StealthNet can scan the live environment and the simulated environment seamlessly (i.e. a scan of ports on the network hardware elements and the simulated StealthNet elements appear the same to the user).

Another set of tools in the StealthNet system that enable live users to devise attack strategies against a LVC representation of the network are “Adaptive Attack Logic”, “Adaptive Attack Proxy”, and “Adaptive Attack Interface”. These tools allow the user to script a sequence of attacks where new attacks can be initiated automatically based on success or failure of the preceding attack. These scripted attacks are useful for comparing the robustness of different tactical network architectures under a given set of attacks.

III. REPRESENTING A TACTICAL NETWORK OVER A DISTRIBUTED T&E INFRASTRUCTURE

This section describes the requirements for representing the full tactical network environment across distributed simulation Instances that are connected via a high speed network infrastructure (either a WAN or LAN) and the technological developments necessary to achieve the requirements.
Often the resources required to robustly test a complex cyber-attack against a System Under Test (SUT) or a tactical mission-based network are located at various, geographically-separated labs. Neither the test resources nor the SUTs can be easily (or economically) moved to a central testing location. Therefore, there is a need for a distributed solution (utilizing a WAN test network) to connect these geographically separated test facilities. The proposed solution is to have multiple Instances of D-SN (each representing a piece of the tactical network under test) running at each of the T&E installations participating in the WAN-based distributed test. The combined effect of the D-SN Instances is to create a seamless representation of the entire tactical network for the distributed SUTs. This section describes this D-SN solution in further detail.

A. Requirements for D-SN

D-SN must comply with three key requirements when representing a tactical network across multiple simulation Instances in a distributed T&E infrastructure:

1. **Preservation of fidelity**: Causality must be maintained such that the results from running on multiple Instances of StealthNet (each representing pieces of the tactical network) are the same as running on a single Instance of StealthNet (representing the entire tactical network). For example, message paths and available link bandwidths through the simulated tactical network must be the same on multiple StealthNet Instances as a single Instance.

2. **Time synchronization**: Time synchronization must be maintained between multiple Instances of StealthNet clocks and the wall clock (be able to run real-time) so that the D-SN system can support high fidelity processing of data from/to external interfaces (to live hardware) thereby supporting LVC environments. Messages must arrive at the same times in a single Instance (representing the entire tactical network) and multiple Instance architectures to support Hardware In The Loop (HWIL) testing (i.e. testing of the Global C2 Systems in a Joint tactical environment).

3. **Minimal bandwidth impact**: Multiple Instance StealthNet-based simulated cyber-attacks must have minimal bandwidth impact on the underlying test infrastructure network connecting the multiple Instances (i.e., the simulated attack cannot become a real attack on T&E network). For example, a DDoS attack initiated at Instance A flooding a simulated node on Instance B in a D-SN architecture cannot consume large amounts of bandwidth in the T&E infrastructure connecting A and B.

These requirements must be achieved while experiencing the natural intermittent latencies of the test network infrastructure connecting the distributed Instances of StealthNet.

B. Decomposition of Scenario for Partitioned Execution

Simulating a large tactical network in real-time often requires realization of the simulated network across multiple StealthNet Instances. These Instances may be colocated and connected by a LAN or located at multiple distributed test sites connected by a WAN. The fidelity and synchronization attributes of the simulated tactical network must be maintained while using minimal bandwidth of the physical network connecting the distributed Instances.

In order to determine how a large scale Army tactical network should be partitioned, several message-load files were obtained from the Army’s tactical Network Integration Event (NIE) exercises. A useful “rule of thumb” was developed to divide the simulated network sections (for each StealthNet Instance) at the junctions of Army tactical echelons. The network for a particular echelon is then represented on one Instance. Figure 2 provides a graphic of this process. In this case, the break or “interconnect” between two StealthNet Instances is made at the Brigade/Battalion echelon with portions of the WNW and WIN-T networks supporting Brigade represented on StealthNet Instance 2 and the WIN-T connections and SRW supporting Battalion (and subordinate Companies) on StealthNet Instance 1.

C. Methods to Minimize T&E Infrastructure Bandwidth for Cyber-Attacks in a D-SN Environment

Types of cyber-attacks represented in D-SN can be classified in two categories: deterministic and non-deterministic.

Deterministic cyber-attacks result in deterministic impact on the remote StealthNet Instance. These attack types have a low dependence on simulation information interchange between the initiating and impacted Instance. Consequently, it is possible to minimize the T&E network in-
fragmentation requirements in these types of attacks. A jamming attack is an example of deterministic cyber-attacks because its attack messages are either responses to wireless signals (in the case of a reactive jammer) or pre-determined (in the case of timer/random seed based jammers). This means that instead of sending the jammer attack message across the infrastructure-link D-SN allows a user to create a “ghost” copy of the jammer node at the remote Instance and only send control messages as shown in Figure 3.

Figure 3. Message syncing for deterministic cyber-attack functionality for jamming attack between StealthNet Instances

Non-deterministic cyber-attacks result in non-deterministic (high entropy) impact on the remote StealthNet Instance due to having a heavy dependence on the network effects of the local StealthNet Instance. DDoS/Malware attacks are examples of non-deterministic cyber-attacks. Their attack messages are actual network packets that must traverse the local network before they can be sent to the remote network. It is not possible to “ghost” these cyber-attacks at the remote Instances to mimic their impact without losing significant fidelity. The remote Instance will lack all of the network effects impacted by the local Instance.

Figure 4 represents a DDoS cyber-attack simulated across two Instances of StealthNet. The Instances are linked through the interconnect-link which can be either a LAN or WAN. StealthNet 2 contains the simulated server network targeted for attack and StealthNet 1 contains the simulated user networkinfected by BOTs launching the attack. The server network is connected to the user network over a single high-delay wired link which crosses through the interconnect-link. The attack proceeds with some hosts in the user network becoming infected with malware and launching a DDoS attack targeting a server in the server network. The goal of the attack is to crash the server by consuming all of its available resources.

In the actual StealthNet Instance 1/Instance 2, T&E infrastructure, lossless compression must be performed on the attack messages (originating in Instance 1) to reduce the load on the interconnect-link. The actual attack messages pass over the LAN/WAN link in compressed form and must be unpacked in Instance 2 to move through the simulated network. These two optimizations (metadata compression of the attack message content and message packing) will allow attacks over the interconnect without significant degradation of LAN/WAN bandwidth.

IV. TIMING SYNCHRONIZATION AMONG D-SN INSTANCES UNDER WAN LATENCIES

StealthNet is a Discrete Event Simulator (DES) and the time synchronization challenges for D-SN are similar to other distributed DES with one major difference. The time synchronization granularity required for D-SN is very high, on the order of milliseconds, to maintain event causality and overall LVC environment fidelity. D-SN also must run over a distributed WAN T&E infrastructure that will have natural intermittent latencies. Unfortunately these latencies, on the order of milliseconds, often approach the latencies being modeled in the tactical network (as simulated tactical messages move from one StealthNet Instance across the test WAN to another portion of the simulated network on a remote StealthNet Instance).

A. Issues with Synchronization of D-SN in a WAN Environment

To guarantee causality, a local StealthNet Instance can only process an event with an event time of $T_{next}$ if the scheduler has already received all remote events with event times $<= T_{next}$. If the local Instance does not wait, any remote events with event times $<= T_{next}$ will be processed at a time later than $T_{next}$, resulting in violation of causality. In such case there is no way to determine or limit how far ahead the simulation clock has advanced when the late event is finally received. This is why a local Instance will always wait until it has received all remote events (from other StealthNet Instances) with event times $<= T_{next}$ before processing a local $T_{next}$ event.

To ensure all remote events have been received all Instances will periodically update each other with their current simulation clocks using time synchronization messages. When a time synchronization message with a simulation clock time of $T_{sync, current}$ is received from a remote Instance, it ensures that no further remote events
with time \( \leq T_{\text{sync\_current\_time}} \) will be received. This allows the receiving Instance to advance and process to the \( T_{\text{sync\_current\_time}} \). These sync messages create a parallelization of the multiple instances allowing them to advance their clocks locally and remain in sync. To increase parallelization, an Instance can process all events with time \( \leq T_{\text{sync\_current\_time}} + T_{\text{lookahead}} \) (also called \( T_{\text{safetime}} \)) with the caveat that \( T_{\text{lookahead}} \) must be \( \leq \) the minimum modeled communication delay between the Instances. This is why determining how to decompose a scenario for partitioned execution is key. It determines the minimum modeled communication delay between the Instances which then determines \( T_{\text{lookahead}} \). The \( T_{\text{safetime}} \) is the processing time boundary to the next remote event and it is used to make sure the local Instance does not advance the local simulation clock by processing events past it. Figure 5 summarizes this process.

The \( T_{\text{sync\_current\_time}} \) in Figure 5 is the current time of the simulation clock on the remote Instance. Once an Instance of D-SN receives a time synchronization message, it can continue processing events up until the new \( T_{\text{safetime}} \), or \( T_{\text{safetime}} = T_{\text{sync\_current\_time}} + T_{\text{lookahead}} \), before syncing up again. In a WAN environment with natural intermittent latencies, there will be an additional delay for sending a time synchronization message to the remote Instances. This delay will cause processing local events, causing an artificial “stop and wait” effect and resulting in a cumulative delay which increases the simulation runtime. If the runtime becomes slower than real time, emulation mode fidelity will be compromised and military hardware being tested in the StealthNet LVC environment may not function properly.

The impact of the additional cumulative synchronization delay on a scenario shown in Figure 6 is from results of a case study where the same scenario ran on D-SN composed of two Instances connected over LAN versus WAN. The \( T_{\text{lookahead}} = 100\text{ms} \) (for the simulation events) with the WAN delay selected randomly from a uniform distribution on a range of 60-140ms. The total simulation time of the scenario is 300s. While the scenario ran faster than real time over a LAN (blue line), the extra cumulative random WAN delays due to waiting for synchronization messages slows the runtime down to 600s (green line) which is twice as slow as real time.

Figure 6. Runtime of the same scenario on D-SN over a LAN vs. WAN

This synthetic delay compromises emulation fidelity as the simulator is processing events twice as slow as real time which makes it unacceptable for emulating environments to test C4I devices.

B. Time Synchronization Message Pipelining

To solve the cumulative synchronization delay issue caused by synchronization messages traversing a WAN, research found that the effects of the delay could be minimized by sending more frequent time synchronization messages. Instead of processing all events up to \( T_{\text{safetime}} \) before sending a synchronization message, StealthNet Instances will send messages continuously and asynchronously in a pipelined fashion. This synchronization message pipelining will cause a parallelization effect between the sync delay and the processing delay. As an Instance continuously receives remote time synchronization messages, it can advance \( T_{\text{safetime}} \) accordingly. The synchronization wait time after an Instance processes its events for the current \( T_{\text{safetime}} \) will be reduced in proportion to the extra synchronization messages.

Instead of waiting until “traditional” \( T_{\text{safetime}} \) before sending a time synchronization message, each Instance will send a pipelined synchronization message every \( T_{\text{lookahead}}/\text{pipeline factor} \) with the current time of their simulation clocks and then continue execution. The purpose of the pipeline factor is to govern how much parallelization is needed for the current scenario and the underlying test infrastructure. The Instances will still only execute events up to the current \( T_{\text{safetime}} \). When receiving a synchronization message, an Instance will update its \( T_{\text{safetime}} \) accordingly \( (T_{\text{sync\_current\_time}} + T_{\text{lookahead}}) \). This increases the number of synchronization messages sent across the interconnect-link by the pipeline factor.
A case study was performed to demonstrate the runtime improvements due to time synchronization message pipelining for D-SN over WAN. The study applied pipelining to the same scenario from the previous subsection and found that pipelining improves runtime performance dramatically as shown in Figure 7. If using a pipeline factor of 5x or more, the effects of the WAN latency are negligible (runs for 5, 10 and 20 overlaid on left blue/green line) when compared to the LAN case. The scenario no longer runs slower than real time when distributed over a WAN. The distributed Instances of StealthNet can synchronize with wall clock and maintain fidelity in emulation mode.

Figure 7: Runtime of the same scenario over WAN using synchronization message pipelining

Figure 8 shows snapshots of a received multicast video stream that traverses through an emulated network when the Instances are connected over WAN with and without synchronization message pipelining. Clearly, the video quality is dramatically improved in the pipelining case. However, the quality of the received video without pipelining is severely comprised due to the extra cumulative WAN delay causing D-SN to run twice as slow as real time, consequentially reducing the emulated network throughput by half.

V. RELATED WORK

D-SN is a capability built upon previous work done in [2] which presented StealthNet as an LVC framework for cyber operation test and evaluation, albeit in a non-distributed context. In [2], the authors demonstrated how the basic StealthNet framework could be used to evaluate the impact of cyber attacks against a Time Sensitive Targeting (TST) mission thread. The core of D-SN is Soft-ware Virtual Network (SVN) technology [3] that makes it possible to represent the communication infrastructure at high levels of fidelity. In this SVN simulation/stimulation environment, actual applications and actual network traffic, such as a mix of sensor data, streaming video, voice communications, chat, collaboration, video web conferencing can be deployed unmodified through the virtual networks. SVN technologies utilize network emulation technology to provide a high fidelity, computationally efficient, and scalable environment for cyber operations. D-SN can interface with existing LVC military network simulation and emulation tools like the Communication Effects Server (CES) [4], the Army’s Brigade level model OneSAF, and the Operational Test Command’s test environment Battle Command Network Integration and Simulation (BCNIS) [5] as well as commercial network simulators.

Other examples of cyber simulation/emulation testbeds include CORONA [6], CPG [7] and Sandia’s LVC approach [8]. CORONA is an enterprise approach for cyber-space testing infrastructure to enable rapidly reconfigurable cyberspace T&E. The CORONA framework was built to support interoperability and reduce setup and sanitation times for the cyberspace infrastructure. This was accomplished by providing a common lexicon and framework of standard interfaces and tools to integrate acquisition systems, networking equipment, emulations, and constructive simulations together. CPG is a cloud based framework to establish a testbed for cyber simulation that allows users to configure multiple virtual environments for simultaneous run of multiple simulations different virtual appliances can be easily deployed. Sandia’s LVC approach is a cyber LVC testbed that combines modeling and simulation capabilities with virtual machines and real devices to represent, in varying fidelity, secure networked information system architectures and devices. They provide “experiment-in-a-box” capability as secure networked information system architectures can be represented on a single computing platform. All the testbeds described in this paragraph are not designed to run across a distributed environment with WAN latencies. It is difficult for these testbeds to scale to large node counts as they rely on the use of virtual machines to sanitize their testbed environment against cyber effects which results in a significant processing overhead as compared to D-SN’s approach of realizing a cyber testbed based on models running in a discrete event simulation framework.

VI. CONCLUSIONS AND FUTURE WORK

The D-SN technology has been developed to allow the creation of a simulated tactical network under various categories of cyber-attack. It is an environment that can be used by the DOD Test and Evaluation community to determine the robustness of various tactical network architec-
ures when attacked during mission scenario testing. The system will represent key mission messages (and other Information Exchange Requirements i.e. video) and the impact of a cyber-attack causing loss or delay of these messages to the mission thread.

The D-SN architecture can be used to represent large scale tactical networks running in real time. Used with a local network for connecting various StealthNet processors, the system can scale easily to represent Brigade-sized networks (1500–2000 elements). Elements in the StealthNet representation of the tactical network include representing RF communications devices (hand held, tactical, and joint radio systems), network elements (fire walls, routers and key data storage devices) and the computer hosts representing key C2 devices.

The D-SN architecture can also be used in a WAN environment linking the DOD laboratories and test facilities. The technology has been developed to represent the state of the simulated mission tactical network through Instances of StealthNet running at each of the DOD test facilities connected to the WAN for synchronization. This synchronization includes:

1. Synchronization of tactical message flow through the simulated tactical network represented by the combined Instances of StealthNet.
2. Synchronization of message arrival times and cyber-attack times (and message volume) across all Instances of D-SN.

For many years, the DOD Models and Simulation communities have used Dead Reckoning techniques to synchronize the movement locations of platforms (both ground based and air fast movers) across multiple simulation nodes. These synchronization requirements were based on the movement speeds of the platforms represented: in most cases on the order of magnitude of seconds. The D-SN technology extends the requirement for Modeling & Simulation (M&S) synchronization by considering the “message movement” in a network as the primary driver changing the state of the simulation space. These movement speeds change the simulation space in milliseconds. The techniques produced by the D-SN technology have advanced the use and fidelity of distributed simulations representing cyber-attack.

As of this writing the StealthNet technology is at Technical Readiness Level (TRL) 6. It has been demonstrated using prototypes of Army Brigade level networks and tactical message-load files seen in many Army training and testing experiments. The technology has been used on several prototype tactical missions (medevac, call for fires on a time-sensitive target and others) with multiple cyber-attacks targeting the mission thread architecture. The technology is being transferred to the Cyber Test Analysis and Simulation Environment (CyberTASE) program where it will be used as the principal simulation in a distributed Joint simulated cyber test mission environment.

VI. REFERENCES

Acknowledgement
The StealthNet project has had support from many people and organizations. The authors wish to acknowledge the unfailing support of the DOD Test Resources Management Center (TRMC) and their Science and Technology Program that sponsored this effort. This project was managed by TRMC’s Netcentric Systems Test, Test Technology Area (NST TTA). Without their technical leadership, focus and resource support, this project would not have become reality.