A virtual ad hoc network testbed

Alex Poylisher*, Constantin Serban, John Lee, Tai-Chuan Lu, Ritu Chadha and Cho-Yu Jason Chiang

Applied Research, Telcordia Technologies Inc.,
One Telcordia Drive, Piscataway, NJ 08854, USA
E-mail: sher@research.telcordia.com
E-mail: serban@research.telcordia.com
E-mail: jolee@research.telcordia.com
E-mail: tedlu@research.telcordia.com
E-mail: chadha@research.telcordia.com
E-mail: chiang@research.telcordia.com
*Corresponding author

Robert Orlando and Kimberly Jakubowski

US Army CERDEC,
Fort Monmouth, NJ 07703, USA
E-mail: Robert.Orlando1@us.army.mil
E-mail: Kim.Jakubowski@us.army.mil

Abstract: Testing of applications for ad hoc networks 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 an ad hoc network. This requires the development of technologies that enable multiple instances of unmodified application software on a set of hosts to communicate via a simulated network that behaves like a real ad hoc network. In this paper, we describe the testbed developed under the *virtual ad hoc network (VAN) testbed* project[1] for testing applications over ad hoc networks, with a special focus on network management applications. The testbed employs *Xen-based virtualisation* to achieve resource scalability. The infrastructure for the testbed provides an integrated platform consisting of *virtual nodes running the actual software under test*, augmented with a *simulated network* environment. Our goal is to enable software testing over large-scale (500–1000 nodes) ad hoc networks using the *VAN testbed*.

Keywords: mobile ad hoc networks; MANETs; virtual ad hoc network; VAN; emulation; software-in-the-loop; SITL; simulation; hybrid testbed; virtualisation; virtual machines; VM; simulated network; shadow nodes.

Reference to this paper should be made as follows: Poylisher, A., Serban, C., Lee, J., Lu, T-C., Chadha, R., Chiang, C-Y.J., Orlando, R. and Jakubowski, K. (xxxx) 'A virtual ad hoc network testbed', *Int. J. Communication Networks and Distributed Systems*, Vol. X, No. Y, pp.000–000.

Biographical notes: Alex Poylisher has been a member of technical staff at Telcordia's Applied Research for the past 13 years. He has a PhD in Computer Science from the University of Warwick, and has worked on research, system engineering and development of various aspects of network, system and service management for both wire line and mobile networks, including policy management, adaptive QoS, information assurance, visualisation and modelling/simulation. His most recent work involves application of cognitive techniques to dynamic QoS and hybrid virtualisation in support of high fidelity software testing for mobile ad hoc networks.

Constantin Serban received his PhD in Computer Science from Rutgers University in 2008. Previously, he received his MS in Computer Science and Engineering from the Polytechnic University of Bucharest, Romania, and BS in Computer Science and Engineering from the same university. His research interests are in policy-based network management for mobile ad-hoc networks, as well as security, dependability, and software engineering aspects of large distributed systems. Currently, he is a Senior Researcher with the Knowledge-Based Systems Department in Applied Research at Telcordia Technologies. He is a member of the IEEE and ACM.

John Lee is a Senior Research Scientist at Telcordia Technologies Inc. in New Jersey, USA. He has been working on future military communication technologies, ad hoc and vehicle-to-vehicle broadcasting and multicasting protocols, and mobility handover technologies. He is a member of Eta Kappa Nu since 1989. He was the recipient of the best paper award in the 1999 IEEE Military Communications Conference. He received his PhD from the Pennsylvania State University.

Tai-Chuan Lu is a Senior Research Scientist at Telcordia Technologies Inc. in New Jersey, USA. He received his MS in Computer Science from the New Jersey Institute of Technology.

Ritu Chadha is Executive Director the Knowledge-Based Systems Department in Applied Research at Telcordia Technologies, where she has been working since 1992. She is the Chief Engineer for Telcordia's Future Combat Systems (FCS) Network Management System, where she is responsible for the design and development of a policy-based network management system for mobile ad hoc networks. She is an industry expert in the area of policy management and has recently authored a book on the subject. She received her PhD in Computer Science from the University of North Carolina at Chapel Hill in 1991. Her research interests include policy-based management, network and service management for IP-based networks, ad hoc networking, and automated reasoning. She is a Telcordia Fellow.

Cho-Yu Jason Chiang is a Director at Applied Research, Telcordia. He received his PhD in Computer and Information Science from The Ohio State University in 2000.

Robert Orlando graduated with BA in Computer Science from Thomas Edison State College and is a Cisco Certified Network Apprentice (CCNA). He is currently employed at US Army Communications-Electronics Research, Development and Engineering Center (CERDEC) where he is pursuing his MSc in Network Information Technologies at Stevens Institute of Technology.

Kimberly Jakubowski graduated from Monmouth University with a BS in Mathematics. She is currently receiving her MS in Electrical Engineering from Stevens Institute of Technology. She has five years of experience in network

management technologies and has held various technical and program lead positions. She was the Network Management Lead on the Communications Planner for Operational and Simulation Effects with Realism (COMPOSER) Advanced Technology Objective (ATO) COMPOSER. She was previously the Network Management Lead on the Tactical Information Technologies for Assured Network Operations (TITAN) ATO. She is currently leading the joint network management interoperability efforts for OSD and is the Co-Chair of the Joint Tactcial Edge Network (JTEN) working group, network management sub-working group.

1 Introduction

Mobile ad hoc networks (MANETs) are characterised 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; since 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, ad hoc network hardware, particularly for the custom-built, 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 ad hoc networks in a physical terrain is very costly in terms of logistics, therefore, the number of tests that can be performed would be much less than adequate.

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

This paper presents the testbed developed under the *virtual ad hoc network (VAN)* project to test applications over ad hoc networks, with a special focus on testing network management applications. The VAN 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 ad hoc network, as if each application instance is running on a real mobile node and the packets are transmitted over a real mobile ad hoc network.

Our main contribution for this endeavour has been to enable point-to-point and broadcast communication between Xen-based virtual hosts via an OPNET-based (The OPNET Simulator, 2009) simulated network. We have developed such capability by introducing a special kind of node emulation in which the IP protocol stack functions are split between Xen-based virtual hosts and simulated nodes, at the IP layer. Further, we have used Layer 2 packet forwarding with MAC address translation between the two parts of the split stack to provide increased performance of the testbed. The testbed achieves resource scalability through Xen-based virtualisation. Finally, to facilitate the

testing of different types of applications, we have also developed some capability for managing the resources of the virtual hosts.

The remainder of the paper is organised as follows: Section 2 discusses related work. Section 3 explains the problem background and relevant technologies. Section 4 describes the network model under consideration. Section 5 presents the architecture and functionality of the proposed testbed, along with an introduction to the associated concepts and objectives. Section 6 shows results with performance data for several application executions. Section 7 concludes the paper with a summary and some future plans.

2 Related work

Related work can be grouped into two main categories:

1 virtualisation

2 hybrid testbeds for wireless networks.

Virtualisation has been applied to operating systems both commercially and in research for nearly thirty years. IBM VM/370 (Borden et al., 1989; Seawright and MacKinnon, 1979) made use of virtualisation to allow binary support for legacy code. VMware family of products, such as, VMware workstation, VMware GSX server, VMware ESX server, etc., virtualises commodity hardware, allowing multiple operating systems to run on a single host. All of these examples implement a *full virtualisation* of (or at least a subset of) the underlying hardware, rather than paravirtualising and presenting a modified interface to the guest operating system. LPARs (Borden et al., 1989), on the other hand, use the *paravirtualisation* approach to build an infrastructure for distributed systems. Xen (Barham et al., 2003) is a high performance resource-managed virtual machine monitor (VMM) (or hypervisor) which provides an idealised virtual machine abstraction, thereby allowing multiple commodity operating systems to share conventional hardware in a safe and resource-managed fashion, but without sacrificing either performance or functionality. Xen can use either paravirtualisation or hardware-assisted virtualisation. Due to its flexibility and performance capabilities, we have used Xen in our testbed.

Hybrid infrastructure (Portoles-Comeras et al., 2006; Raychaudhuri et al., 2005; White, 2002; Zheng and Ni, 2003, 2002; Zhou et al., 2006) have proved to be useful in validating networking techniques and conducting large scale experiments for wired/wireless environments. Emulab/Netbed (a descendant of Emulab) (Guruprasad et al., 2005; Hibler et al., 2008) is a network testbed that integrates emulators, simulators and live networks into a common framework under a common user interface. Experiments on Emulab/Netbed testbed can combine real elements with simulated elements, each modelling different portions of a network topology in the same experimental run. The testbed can be used for comparisons of simulated, emulated, and wide-area scenarios. The experimental testbed for research enabling mobility enhancements (EXTREME) (Portoles-Comeras et al., 2006) provides an experimentation facility that supports the testing of networking algorithms and technologies for wireless environments in a close-to-real scenario. Its design has been inspired by Emulab. EMWIN/EMPOWER (Zheng and Ni, 2003, 2002) is an IP-based scalable framework for mobile wireless emulation. With EMWIN, the mobility of the target mobile wireless

network with a number of mobile nodes can be faithfully emulated in a wired network environment. EMPOWER is capable of assisting the study of both wired and wireless network protocols and applications. The open access research testbed for next-generation wireless networks (ORBIT) (Raychaudhuri et al., 2005) is a radio grid testbed that has been developed for scalable and reproducible evaluation of next-generation wireless network protocols. The ORBIT testbed consists of an indoor radio grid emulator for controlled experimentation and an outdoor field trial network for end-user evaluations in real-world settings. WHYNET (Zhou et al., 2006) is a large-scale hybrid testbed for heterogeneous wireless technologies (MANET, Wireless LAN, 3G Cellular, Sensors, UWB, etc.) that combines the realism of physical testing with scalability, flexibility and repeatability of simulations.

3 Background

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

3.1 Network simulation and network emulation

Two of the most commonly used experimental techniques for the design and validation of new and existing networking ideas and technologies are network simulation and network emulation. In *network simulation*, a program models the behaviour of a network either by reproducing the interactions between the different network entities (hosts/routers, data links, packets, etc) using mathematical formulas, or by replicating behaviour of real network components. Network simulation provides a repeatable and controlled environment for rapid prototyping and experimental evaluation. *Network emulation* (Hibler et al., 2008; Portoles-Comeras et al., 2006) is a hybrid approach that combines real elements of a deployed networked application, namely, the end hosts and protocol implementations, with synthetic, simulated, or abstracted elements, namely, the network links, intermediate nodes and background traffic. A fundamental difference between simulation and emulation is that while the former runs in virtual simulated time, the latter must run in real time. Another important difference is that it is impossible to have an absolutely repeatable order of events in any emulation, due to its real-time nature and often a physically-distributed computation infrastructure. *Hybrid testbeds* perform integrated network experimentation by combining real and/or emulated elements with simulated ones.

3.2 Software-in-the-loop simulation

Software-in-the-loop (SITL) simulation is a methodology that utilises a simulation-based approach to evaluate the effectiveness and scalability of real software as it is deployed in the field (Chiang et al., 2006). This methodology combines the use of unmodified compiled code from a real application with the simulation model. 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. This methodology also eliminates the need for

deriving and developing a model for the real software in the simulator, as the in-the-loop software integrates easily with the simulation model. To be effective, the SITL solution needs to be platform-independent and neutral to the choice of the simulation engine. In our experiments, we model the MANET in a *simulator* and use Linux-based applications as the *SITL*.

3.3 Virtualisation

In computing, *virtualisation* is a broad term that refers to the abstraction of computer resources. Virtualisation makes it possible to run multiple operating systems (OSes) on the same computer at the same time. Virtualisation can be grouped into two main types, i.e., platform virtualisation and resource virtualisation. *Platform virtualisation* involves the emulation of whole computers, while *resource virtualisation* involves the emulation of combined, fragmented, or simplified resources. Platform virtualisation is performed on a given hardware platform by *host software* (a *control program*), which creates an emulated computer environment, a *virtual machine*, for its guest software. The *guest software*, which is often itself a complete OS, runs just as if it were installed on a stand-alone hardware platform. Typically, many such virtual machines (VMs) are emulated on a single physical machine, where the number of the VMs is limited by the host's hardware resources. Generally, there is no requirement for a guest OS to be the same as the host one. Three main approaches to platform virtualisation are: *full virtualisation*, *hardware-assisted virtualisation* and *paravirtualisation*. In the first, the virtual machine emulates enough hardware to allow an unmodified guest OS (one designed for the same CPU) to be run in isolation (e.g., Virtual PC, QEMU). In the second, the hardware provides architectural support that facilitates building a virtual machine allowing unmodified guest OSes to be run in isolation (e.g., Linux KVM, Xen, etc.). The second approach is an optimisation of the first. In the third, the virtual machine does not necessarily emulate hardware, but instead (or in addition) the VMM offers a special API that can only be used by the modified guest OS (e.g., Denali, Xen). In this paper, we present the use of Xen paravitualisation approach for hosting several *Linux-based* guest OSes on the same physical box, to run several instances of the network management applications.

4 Network model

The network model used in the *VAN testbed* implements the functionality of a network layer (OSI) and below. The network layer in our model consists of two sub-layers, i.e., the *mobile network layer* and the *mobile link layer*. In this model, packets originate at the IP layer. The IP packets are transferred to the *mobile network layer* and subsequently handed down to the *mobile link layer*. The mobile link layer is responsible for modelling TDMA transmission between two neighbouring nodes within radio range. The mobile network layer is responsible for packet routing and end-to-end (multi-hop) forwarding. IP packets are encapsulated within the mobile network layer packets at the source node and are de-capsulated at the destination node. Packets arriving at intermediary nodes are forwarded at the mobile network layer level without reaching the IP layer. In this model, the IP layer is introduced to offer compatibility with the IP based networks, and it functions as a wrapper for the mobile network layer.

The network model supports two types of traffic:

a unicast traffic, destined for a single destination node

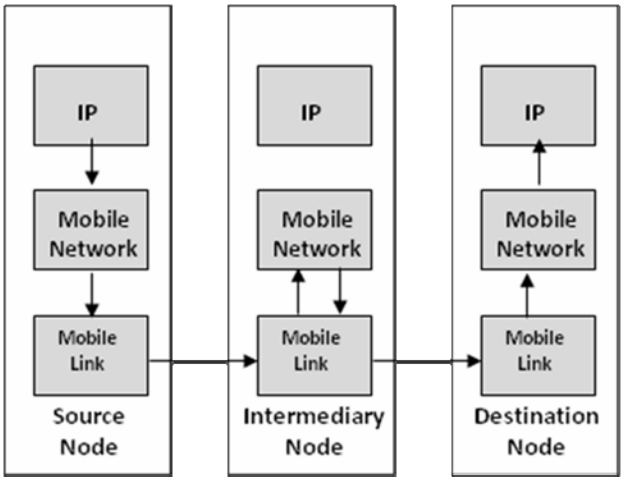
b broadcast traffic, destined to all the nodes within radio range and which are assigned to the same mobile subnet as the source node.

Nodes with multiple radio interfaces belong to multiple subnets; such nodes play the role of gateways.

Using the above model, we have developed several network scenarios designed to replicate the typical numbers of interacting nodes, subnet and channel allocations, as well as real life mobility trajectories for these nodes. The numbers of mobile nodes that are part of a network scenario, as well as their maximum traffic rates have been limited by the *VAN testbed* constraint to have the simulation run no slower than real time. A more detailed description of some network scenarios is provided in Section 6 along with the applications tested over these scenarios.

Figure 1 shows the model of a network node, and the flow of packets between its layers from a source node to a destination node.

Figure 1 Mobile node model and cross-layer packet flow



5 A virtual ad hoc network (VAN) testbed

Testing and evaluation of real applications over dynamic MANETs pose a significant challenge. One approach to address this challenge is to conduct testing over a testbed that behaves like a real MANET. We propose a *Xen-based virtualisation* (Barham et al., 2003) of the MANET. Accordingly, we have developed a *VAN testbed* that applies the SITL simulation methodology to integrate a simulated MANET with the *in-the-loop* real network management application instances, running on Linux-based VM. The testbed virtualises network resources on physical host machines. With Xen virtualisation, a thin software layer known as the *Xen hypervisor* is inserted between the server's hardware

and the operating system. It provides an abstraction layer that allows each physical server to run one or more *virtual servers*, effectively decoupling the operating system and its applications from the underlying physical server. The Xen hypervisor is a powerful open source industry tool for virtualisation. It can use *paravirtualisation* as well as *hardware-assisted full virtualisation*. It offers a powerful, efficient, and secure feature set for virtualisation of several mainstream CPU architectures, and supports a wide range of guest operating systems including Linux, Solaris, various BSD variants and Windows (unmodified). A Xen system is structured with the Xen hypervisor as the lowest and most privileged layer. Above this layer are one or more guest OSes, which the hypervisor schedules across the physical CPUs. The first guest operating system, called *domain 0 (Dom0)*, is automatically booted after the hypervisor. Additional guest OSes, called *domain Us (DomUs)*, are started from Dom0. DomUs are managed from Dom0.

5.1 Acronyms, concepts and objectives

This section provides a collection of terminologies, concepts and objectives that are essential for describing and implementing the architecture and functionality of the proposed *VAN testbed*.

Definition 5.1 *Network virtualisation* represents the process for combining hardware and/or software to create a *virtual network* that behaves like a real network. A *VAN* virtualises a MANET.

Definition 5.2 An *emulated node* represents a node that imitates a real node, emulates its protocol implementation, and runs its operational applications. As part of this *emulation*, the protocol stack of an emulated node can be split into two parts at some layer of its protocol stack, where the upper part of the protocol stack is implemented by actual OSes, e.g., Linux and Windows, and the bottom part is implemented by simulation models of lower layers.

Definition 5.3 *A virtual machine (VM)* represents a node in the *VAN testbed*, hosted on a DomU, which can run an instance of the *emulated node* including only the upper part of the protocol stack.

Definition 5.4 *A simulated network* represents a collection of software components that model the functionality of a real network and its devices. For example, a *simulated network* can be implemented using a network simulation tool like OPNET.

Definition 5.5 *A split protocol stack* represents a protocol stack where the higher layers of the stack are implemented in the *VM* and the lower layers of the stack are implemented by the *simulated network*. The point at which the stack is split is called the *splitting layer*.

Definition 5.6 *A shadow node* represents a node in a *simulated network* that simulates the functionality of certain ad hoc network communication protocols. In this paper, the shadow node is assumed to implement the part of split protocol stack at and below the *splitting layer*.

Our *goal* has been to develop a testbed that meets the following *objectives*.

Objective 5.1 The testbed shall support *emulation* of a MANET by using a *SITL simulated network*.

Objective 5.2 The testbed shall support *emulated nodes* where the part of the protocol stack above and including the *splitting layer* runs on a *virtual machine*, while the part of the protocol stack below and including the *splitting layer* runs on a *shadow node*.

Objective 5.3 The testbed shall support IP as the *splitting layer*.

Objective 5.4 The testbed shall support unicast transmission over the *simulated network*.

Objective 5.5 The testbed shall support testing of any unmodified, distributed third party application running over *simulated networks*.

Objective 5.6 The testbed shall allow *remote management* of its resources.

5.2 Core functionality

Our goal is to create a MANET testbed that enables testing and evaluation of any software application on supported operating systems for any MANET scenario, without necessitating changes to the application under test. The core functionality of the testbed involves the following major operations.

1 Transparent forwarding – the testbed facilitates transparent communication among *VMs*, using IPv4 addresses. Messages exchanged between applications running on different *VMs* will be forwarded through the simulator machine, without the applications being aware of them.

2 Local stack management – the testbed provides a mechanism for the applications to retrieve information available in the *simulated network* at their corresponding shadow node. Typical information provided includes the current mobile node coordinates and radio characteristics, such as the SNR. This communication is meant to assist applications that are designed to interact with the layers of the *split protocol stack* below the *splitting layer*. In the current implementation, this interaction is SNMP-based. The descriptions of the local stack management and its usage are beyond the scope of this paper.

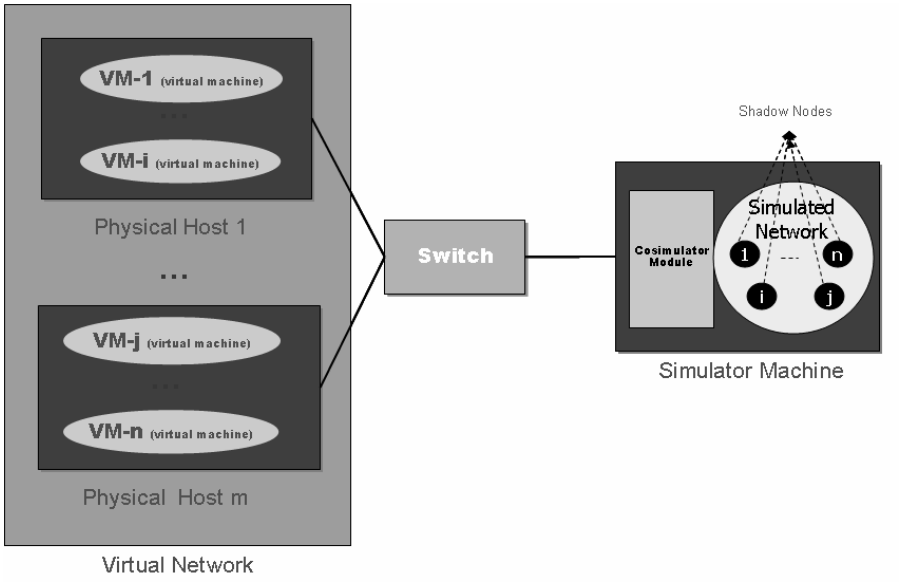
3 Remote testbed management – the testbed allows remote management of applications through interfaces that are different from those used for transparent forwarding.

5.3 Architecture

The *VAN testbed* supports the *emulation* of a MANET by using a *SITL simulated network*. Accordingly, our testbed can currently integrate *emulated nodes* and a *simulated network* into a common framework under a common user interface. An *emulated node* in our testbed employs a *split protocol stack*, with IP as the *splitting layer*. Experiments on our testbed can therefore, combine *VM* with *shadow nodes*, in the same experimental run. This section presents the system architecture of the *VAN testbed*.

Figure 2 provides an overview of the physical architecture of the entire testbed. The following is a brief description of its two main functional components.

Figure 2 Physical architecture of the VAN testbed



Simulator machine: This is a physical machine that is dedicated to running the *simulated network*. The *simulated network* hosts several *shadow nodes* (1..n) and comes with a cosimulator module (COSIM) that integrates it with the rest of the testbed (virtual network). This machine also runs the WinPCap daemon and the Windows firewall. Currently, the testbed uses OPNET Modeller 11.5 for the simulated network. This is executed on a dual-core, two-processor machine with 2G of RAM per 2 GHz processor, running Windows.

Virtual network: The testbed uses a Xen-based virtualised network to host the applications to be tested. The entire virtual network is hosted on several Intel quad-core, eight CPU, 8G physical host machines (1..m). On each such physical host machine, Xen hypervisor is installed at the lowest layer; NetBSD-current is installed as Dom0 and Linux (Fedora Core 8) as DomUs. There are several *VM* (DomUs), named VM-1 through VM-n, that are running on these physical machines (1..m), each with 256M of RAM. Each *VM* (DomU) is on its own subnet (in the 10.0 space), and can indirectly talk to other *VM* (DomUs). Each VM (DomU) is provided with a separate management interface, using an address from the 10.128 address space, through which graphical applications can also be run. Each Dom0 has two interfaces; one (Dom0-bnx0) connects it with the rest of the testbed through an internal LAN and the other (Dom0-bnx1) provides access to the outside world. For each *VM* ($DomU_i$, $1 < = i < = n$), Xen creates two new pairs of *connected virtual Ethernet interfaces*, where one end of each of the two pairs, i.e., ($DomU_i$-eth0) and ($DomU_i$-eth1) are within the $DomU_i$ and the other end, i.e., (Dom0-xvifi.0) and (Dom0-xvifi.1) exist within Dom0. There are (2n) bridges (Br_i, $1 <= i < = 2n$) in Dom0, one for each pair of connected interfaces.

Figure 3 and Figure 4 provide overviews of the high-level and detailed architecture (bridges, interfaces, sub-networks, etc.) of the virtual network, as hosted on a single physical machine, i.e., for the base case of the virtual network with a single physical host machine.

Figure 3 High-level architecture of a single physical host in the virtual network

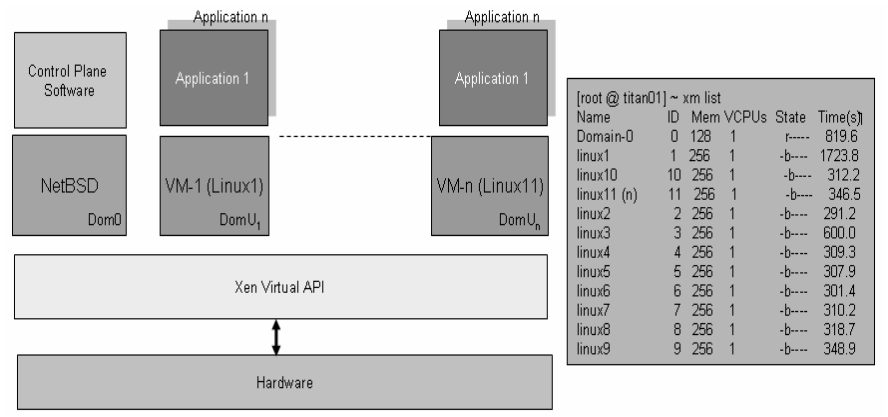
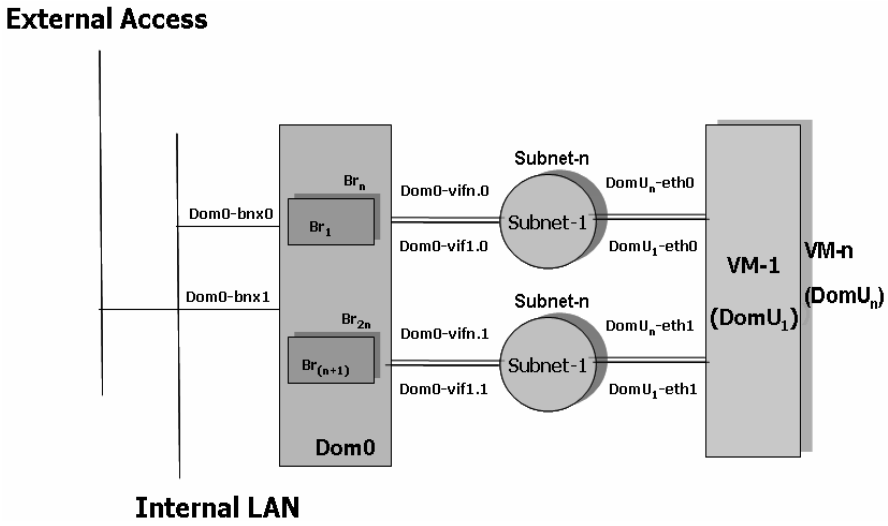


Figure 4 Detailed architecture (interfaces and bridges) of the virtual network

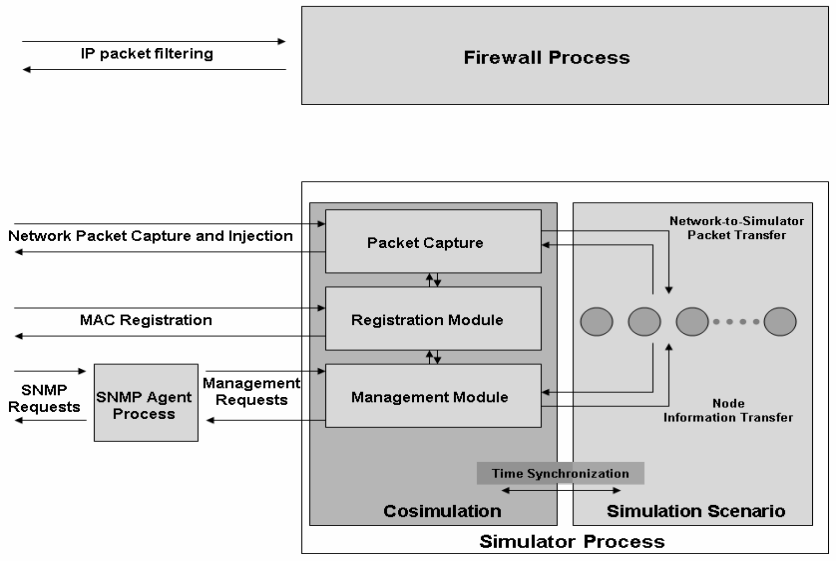


5.3.1 The simulator machine and relevant processes

The main task of the simulator machine is to execute the simulation scenario corresponding to the desired network behaviour, and to route the IP communication between the *VM* through this simulation. In order to enable this functionality, the

simulator machine executes a number of processes and services, as depicted in Figure 5, including the simulation process, the SNMP agent process, and the firewall process.

Figure 5 Processes and servers on the simulator machine



The *simulation process* is composed of the following subcomponents:

- The *simulation scenario* models a number of nodes and their Layer 3, 2, and 1 communication stacks. The scenario executes for a given period of time, simulating given trajectories for each individual node.
- The *COSIM* provides a framework for launching and controlling the execution of the simulation. The main purpose of the cosimulator is to transfer real communication packets into the simulation. The cosimulator achieves this through the following components:
  - The *packet capture component* is a server based on PCap that listens to, and sends out Layer 2 packets originating in, and destined for the virtual nodes. The packet capture component is also responsible for transferring packets to and from the simulator.
  - The *registration module* is a server that maintains a mapping between the IP address of the VM and the physical MAC addresses of the machine hosting them. Physical hosts hosting VM register to this module prior to starting the simulation scenario. The MAC address corresponding to virtual machine is used by the packet capture component to correctly route the packets to their destination.
  - The *management module* is a server that exports network information that is modelled in the simulation to interested parties. Typically, virtual nodes are querying information related to their corresponding shadow simulation nodes.

The *COSIM* is also responsible for advancing the simulation scenario in real time, or at a pace suitable for the virtual nodes.

- The SNMP agent process is a SNMP agent that serves as a front end to the management module.
- The firewall process is responsible for dropping the Layer 3 (IP) packets that are intercepted by the packet capture component at Layer 3, in order to prevent possible duplication and improve performance. The firewall drops all incoming IP packets arriving at the simulation host but destined for an IP address other than that of the Simulator Machine.

5.3.2 Transparent forwarding

A virtual hub-and-spokes network is set up between *VM*s with the simulator standing-in as the hub. Messages can be exchanged between applications running on any two *VM*s (DomUs), hosted on the same or different physical machine(s), via the simulator machine. Indirect (virtual) point-to-point or broadcast communication between any two VMs (DomUs) is accomplished through the use of packet filter (PF) rules on Dom0, a packet capture daemon, (WinPCap), and Windows firewall on the simulator machine along with the MAC address registration process involving *VM*s and physical hosts. All *VM*s register the mapping between their IP address and the MAC address of the physical machine hosting the *VM* with the WinPCap daemon. MAC address registration and translation are required as the WinPCap daemon processes the incoming and the outgoing frames at Layer 2 of the protocol stack, thereby routing at the MAC layer and not at the IP layer. PF rules are used on Dom0, to filter out the incoming interfaces, such that all outgoing traffic from any *VM* (DomU) destined to another *VM* (DomU) is first forwarded towards the simulator machine. The WinPCap daemon of the COSIM on the simulator machine subsequently routes copies of the frames via the *shadow nodes* in the OPNET Simulator, while the Firewall drops the actual traffic. Finally, the WinPCap daemon, after receiving back the frames from the OPNET simulator, inserts the appropriate destination MAC address corresponding to the IP address of the destination *VM* (based on the dynamic mapping table) in each frame, and then sends them to the destination *physical host*.

Figure 6 provides the details of this information flow. This figure describes the travel path of a frame (Layer 2 packet flow) from VM-1 of physical host 1 to VM-n of physical host m. The path consists of four segments, labelled (1) through (4). Segment (1) shows packet flow between physical machine hosting VM-1 and the simulator machine hosting the COSIM & the simulator. Segment (2) shows packet flow between the WinPCap daemon (inside COSIM) and the OPNET simulator inside the simulator machine. Segment (3) shows packet drop resulting from packet flow between the WinPCap daemon (inside COSIM) and the firewall. Segment (4) shows packet flow between the simulator machine and the physical machine hosting the destination *virtual machine*, i.e., VM-n. Black dotted lines point to packet contents at different segments of the path. In this path, the content of the frame changes only in segment (4) due to MAC address substitution inside the WinPCap daemon of the COSIM.

Figure 6 Information flow between two virtual machines for transparent forwarding

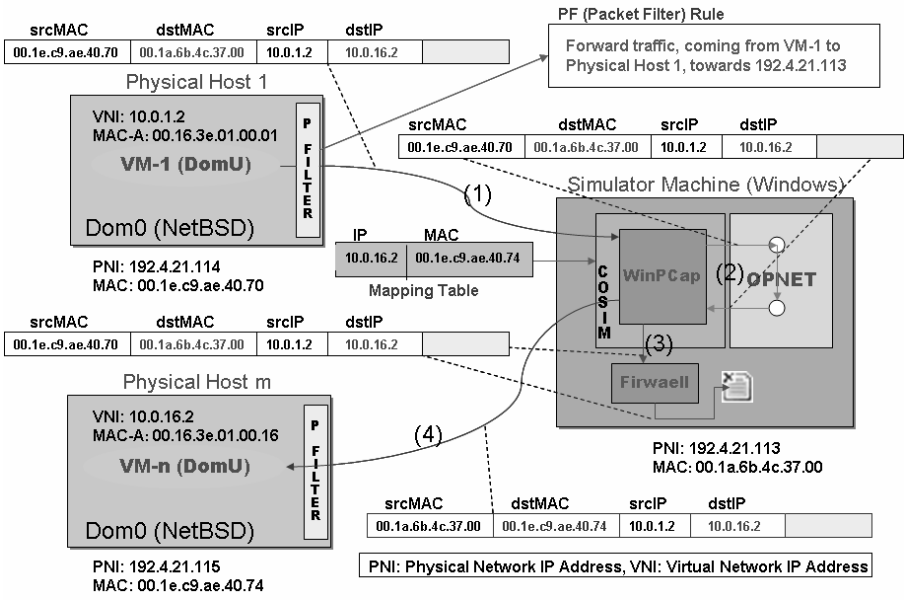
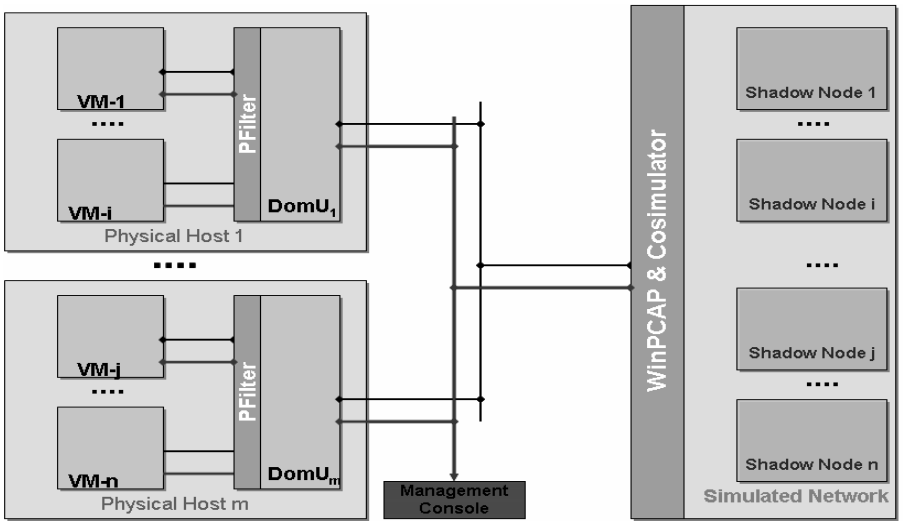


Figure 7 Overview of VAN testbed management



5.3.3 Remote testbed management

The *VAN testbed* allows users to control and manage its resources. Users can access *VM*s and the simulator machine and manage the process of testing. Each *VM* provides two network interfaces, one for *transparent forwarding*, and the other for management purposes. The management interface is different from the network interface used for *transparent forwarding*. It is used to access the *virtual machine* externally for command line interface, file transfer, application installation, graphical user interface (GUI) support, etc. Figure 7 displays the two interfaces.

6 Van testbed applications and results

In order to assess the effectiveness and performance of our testbed implementations, we have run a number of off-the-shelf communication-oriented applications. These applications fall into the following categories.

6.1 Network management applications

Dynamic re-addressing and management for the army (DRAMA, Chadha et al., 2004; Chiang et al., 2006) is a policy-based MANET management system. DRAMA is composed of distributed autonomous agents, which organise themselves into a hierarchy to disseminate policies across the network and collect management information according to policies.

DRAMA represents a good candidate for execution over our VAN testbed due to several of its salient features. First, DRAMA's management hierarchy is self-organising, and it reflects the current network connectivity. Second, DRAMA has small bandwidth requirements, making it suitable for the type of mobile ad-hoc networks that we simulate. Third, DRAMA uses both broadcast communication (for the discovery of one-hop neighbours), and unicast messages for establishing the management hierarchy and other activities, thus, providing a comprehensive and realistic application designed to run in MANET environments.

In order to evaluate DRAMA over the *VAN testbed*, we have used a simulated network composed of 21 nodes. The testbed had been configured to execute on 3 physical machines, each hosting seven *VM* (DomUs), each hosting a Linux FC8 OS. Each *VM* (DomU) has been configured to execute a DRAMA agent. The network simulation scenario had been configured with 21 nodes, each corresponding to a *VM* (DomU). These nodes were organised into five subnets reflecting the relative node movements, where an entire subnet moves in a single compact formation. Figure 8 shows the animation display providing the real time position of each node in the simulation. Subnets are depicted as compact groups of red nodes. These subnets move along the green arrowed lines. Subnets 2 and 3 are each composed of seven nodes advancing eastwards; subnets 4 and 5 are composed of three nodes each, initially advancing westwards. Node 1 represents a stationary node assigned to subnet 1. One node in each of the subnets 2–5 is also assigned to subnet 1, to act as a gateway for its corresponding subnet. Red circles depict the radio range of node 1, as well as the ranges of subnets 4, and 5 respectively.

Figure 9 displays the DRAMA management hierarchy formed as a result of messages exchanged over the *VAN testbed* during the network scenario. The Drama instance

executing on node one is depicted in red, representing the root of the hierarchy. The blue nodes directly connected to node one represent the gateways for subnet 2 and 3, while the green nodes under them represent the rest of the nodes in subnets 2 and 3. The hierarchy connectivity reflects the direct communication range between various nodes, as observed in the simulation animation display. The two separate, unconnected hierarchies at the bottom of this display reflect the partitioning of the network, due to subnets 4 and 5 being out of range. In the course of the scenario, when subnets 4 and 5 move within radio range of subnets 2 and 3 respectively, DRAMA's periodic broadcasts get forwarded, and the DRAMA hierarchy becomes fully connected.

Figure 8 21-node network scenarios

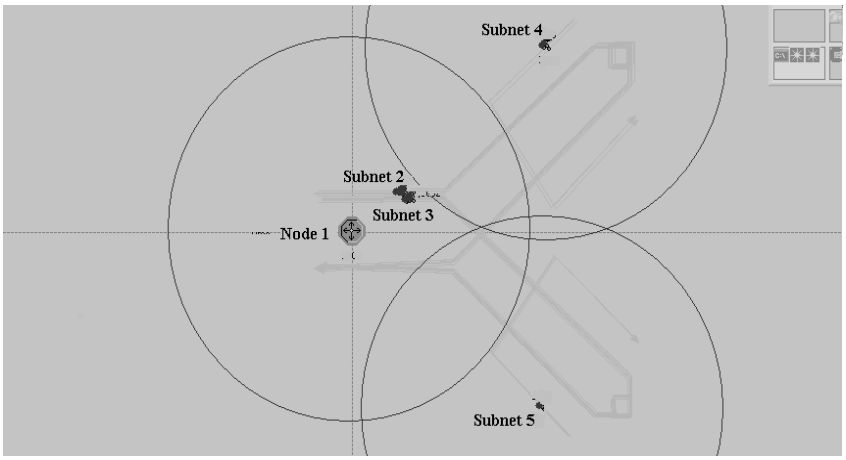
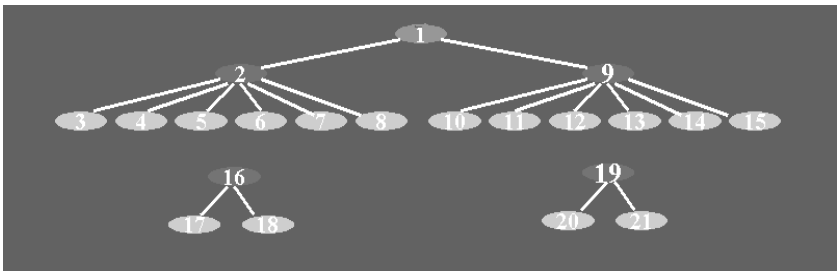


Figure 9 DRAMA management hierarchy



6.2 Network tools

In order to assess the modeled *network connectivity*, *communication latency*, and *drop rates*, we have used the standard *ping* command between different pairs of *VM*s (DomUs). Ping request and reply packets, traveling through the simulated network, become subject to the modeled latency and drop rates. When node A pings node B, and the two nodes are disconnected, the ping output reflects no *ping echo reply* packets arriving back at node A, and the reported packet drop rate becomes 100%. When a

multi-hop path between nodes A and B can be found, *ping echo reply* packets start arriving at node A, displaying a latency proportional with the number of hops between the two nodes. When the two nodes are in radio range of each other, the latency is minimal and the packet drop rate becomes zero.

Figure 10 shows the result of executing ping when two nodes transition from a multi-hop route into direct radio communication. The latency of communication decreases from about 500 ms to 200 ms RTT, thus, accurately reflecting a roughly 100 ms modeled latency for a small packet transmission between two neighbouring nodes.

Figure 10 Ping command output

```
64 bytes from linux65 (10.0.65.2): icmp_seq = 19 ttl = 62 time = 506 ms
64 bytes from linux65 (10.0.65.2): icmp_seq = 20 ttl = 62 time = 414 ms
64 bytes from linux65 (10.0.65.2): icmp_seq = 21 ttl = 62 time = 427 ms
64 bytes from linux65 (10.0.65.2): icmp_seq = 22 ttl = 62 time = 472 ms
64 bytes from linux65 (10.0.65.2): icmp_seq = 23 ttl = 62 time = 206 ms
64 bytes from linux65 (10.0.65.2): icmp_seq = 24 ttl = 62 time = 226 ms
```

6.3 Multimedia applications

In order to evaluate the behaviour of multimedia applications over our model *simulated network*, we have run an off-the-shelf *VIC* program over two *VMs* in our testbed. *VIC* video tool is a real-time, multimedia application for video conferencing over the Internet, developed at UC Berkley/LBNL. VIC represents a good multimedia test application as it offers fine control over the amount of information sent over the network, and the type of transfer protocol employed, thus, becoming suitable for use in mobile ad-hoc networks. Additionally, it offers a free, open source implementation for most major operating systems.

6.4 Benchmarking applications

In order to evaluate the correctness of our *VAN testbed* solution, we have attempted different types of communication (unicast, network broadcast, local broadcast traffic,) as well as different protocols, such as TCP, UDP, ICMP, and raw IP datagrams successfully.

In order to evaluate the overhead introduced by our solution, we have tried to separate the latency and bandwidth factors modeled by the network scenario from the latency and bandwidth caused by the testbed. We have achieved this separation by creating an ideal network model that immediately forwards every single packet to its destination, without introducing any delay and drop rate. Figure 11 presents the average latency for packets transmitted between two *VM* (DomUs) in the presence of moderate traffic (up to 50 Mbps). The two *VM*s (DomUs) were located on different Dom0s, and the latency was averaged over 10,000 packets.

Under such conditions, the observed overhead is between 130 and 220 microseconds. This latency is due to the following components:

1 t_v, *VM* (DomU) to Dom0 latency (at both the sender and receiver sides)

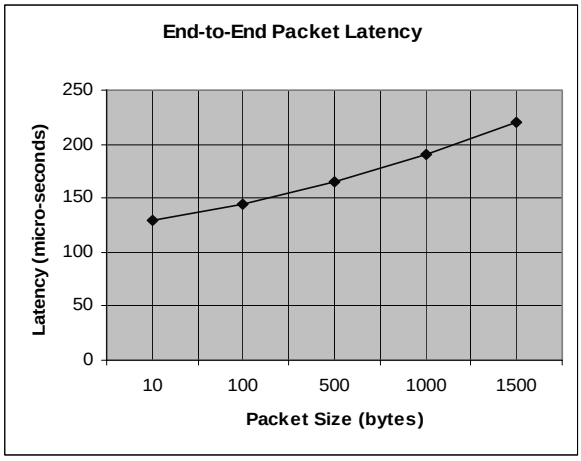
2 t_r, Dom0 to Simulation-process latency (at both sender to simulation and simulation to receiver sides, including two real Ethernet transmissions).

The measured t_v component of the latency amounts to a value of about 25 microseconds, thus, t_r dominates the overall latency.

In a different series of experiments, we have measured the overhead latency for different traffic loads. In addition to the moderate traffic above, the latency measured for 1,000-byte packets at the rate of 80 Mbps was 330 microseconds, and at the rate of 120 Mbps, the latency was 510 microseconds.

Even though the above absolute values might indicate a non-negligible latency, we believe that these values are relatively insignificant compared to the latencies experienced or anticipated in the MANETs under evaluation.

Figure 11 VAN testbed latency overhead



7 Conclusions

We have described a hybrid testbed that employs the SITL simulation methodology for testing applications that can run on real MANETs, addressing the feasibility and scalability related challenges of testing and evaluation over MANETs. We have developed a Xen-based virtualised architecture for this hybrid testbed, where the applications run on multiple *VM* and can communicate via the *simulated network*. The testbed facilitates message exchanges between the virtual and simulated worlds. We have implemented indirect point-to-point and broadcast communication through the use of PF rules, a packet capture daemon and a MAC-address based registration & translation mechanism. The testbed provides facilities for remote management of its resources. We have conducted experiments in the testbed; results demonstrate the applicability of the testbed for testing such applications. Preliminary results indicate that in the context of

MANETS, the virtualised architecture can be advantageous as it not only reduces hardware considerably but also meets or exceeds our performance expectations.

Our future plan is to integrate the current *VAN Testbed* with emulated networks and real network devices in a lab environment. Our eventual goal is to establish benchmarks for running large-scale experiments on performance and QoS measurements in virtualised environments.

Acknowledgements

We would like to thank the Office of the Secretary of Defense (OSD) and US Army CERDEC for supporting the work and providing valuable feedback.

References

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A. (2003) 'Xen and the art of virtualization', *Proceedings of SOSP'03*, New York.

Borden, T.L., Hennessy, J.P. and Rymarczyk, J.W. (1989) 'Multiple operating systems on one processor complex', *IBM Systems Journal*, Vol. 28, No. 1, pp.104–123.

Chadha, R., Cheng, Y.H., Chiang, C-Y.J.G.A., Levin, G. and Tanna, H. (2004) 'Policy-based mobile ad hoc network management, policies for distributed systems and networks', *Proceedings of 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04)*.

Chiang, C.J., Demers, S., Gopalakrishnan, P., Kant, L., Poylisher, A., Cheng, Y-H., Chadha, R., Levin, G., Li, S., Ling, Y., Newman, S., LaVergne, L. and Lo, R. (2006) 'Performance analysis of DRAMA: a distributed policy-based system for MANET management', *Proceedings of IEEE MILCOM 2006*, Washington, DC.

Guruprasad, S., Ricci, R. and Lepreau, J. (2005) 'Integrated network experimentation using simulation and emulation', *The 1st International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM '05)*, pp.204–212, Trento, Italy.

Hibler, M., Ricci, R., Stoller, L., Duerig, J., Guruprasad, S., Stack, T., Webb, K. and Lepreau, J. (2008) 'Large-scale virtualization in the Emulab network testbed', *Proceedings of the 2008 USENIX Annual Technical Conference*, Boston, MA.

Portoles-Comeras, M., Requena-Esteso, M., Mangues-Bafalluy, J. and Cardenete-Suriol, M. (2006) 'EXTREME: combining the ease of management of multi-user experimental facilities and the flexibility of proof of concept testbeds', *Proceedings of the 1st International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM '06)*, Barcelona, Spain.

Raychaudhuri, D., Seskar, I., Ott, M., Ganu, S., Ramachandran, K., Kremo, H., Siracusa, R., Liu, H. and Singh, M. (2005) 'Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols', *Proceedings of IEEE Wireless Communications and Networking Conference*, pp.1664–1669.

Seawright, L. and MacKinnon, R. (1979) 'VM/370 – a study of multiplicity and usefulness', *IBM Systems Journal*, pp.4–17.

The OPNET Simulator (2009) Available online at http://www.opnet.com/.

White, B.L.J.S.L.R.R.G.S.N.M.H.M.B.C.J.A. (2002) 'An integrated experimental environment for distributed systems and networks', *Proceedings of the 5th Symposium on Operating Systems Design & Implementation (OSDI)*, pp.255–270, Boston, MA.

Zheng, P. and Ni, L.M. (2002) 'EMWIN: emulating a mobile wireless network using a wired network', *Proceedings of the 5th ACM International Workshop on Wireless Mobile Multimedia (WOWMOM '02)*, pp.64–71, Atlanta, GA.

Zheng, P. and Ni, L.M. (2003) 'EMPOWER: a network emulator for wireline and wireless networks', *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*.

Zhou, J., Ji, Z., Varshney, M., Xu, Z., Yang, Y., Marina, M. and Bagrodia, R. (2006) 'WHYNET: a hybrid testbed for large-scale, heterogeneous and adaptive wireless networks', *Proceedings of the 1st International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, Los Angeles, CA.

Notes

1 The research reported in this document/presentation was performed in connection with contract number W15P7T-08-C-P213 with the US 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 US Army CERDEC, or the US Government unless so designated by other authorised documents. Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof. The US Government is authorised to reproduce and distribute reprints for government purposes notwithstanding any copyright notation hereon.