IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 3, 2018, accepted November 6, 2018, date of publication November 20, 2018,
date of current version December 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2882410

Development of an Architecture for a Cyber-Physical Emulation Test Range for Network Security Testing

AIDAN F. BROWNE[1], (Senior Member, IEEE), STACEY WATSON[2], AND WESLEY B. WILLIAMS[1]
[1]William States Lee College of Engineering, The University of North Carolina at Charlotte, Charlotte, NC 28223, USA
[2]College of Computing and Informatics, The University of North Carolina at Charlotte, Charlotte, NC 28223, USA

Corresponding author: Aidan F. Browne (aidanbrowne@uncc.edu)

ABSTRACT Although a number of network emulation tools exist, they vary on the level of fidelity of the emulation. Furthermore, most of all emulate a network completely virtually, not allowing for hardware-in-the-loop as part of the emulation. This paper presents an architecture for a cyber-physical emulation test range that operates at high fidelity and allows for incorporation of hardware-in-the-loop with no customization required. The system, which is built on top of VMware, allows for emulation of any network providing that all nodes can either be virtualized or physically attached to the system. Network testing, including penetration testing, can be performed at a high level of fidelity.

INDEX TERMS Emulation, platform virtualization, Ethernet networks, system testing.

I. INTRODUCTION

There are numerous reasons, such as performing a non-invasive penetration test, that it is desirable to be able to emulate a network virtually [1]. This presents a challenge when some of the devices on the network are not able to be virtualized (e.g. because of a custom operating system) [2]. Simulating those devices as a low-integrity node within the virtual emulation preserves the existence of the node, but significantly reduces the integrity of the emulation. The research described here set out to create an architecture that would allow such devices to be physically connected to the hardware on which the emulation is running, and seamlessly stitch them in as nodes in the virtual emulated network (Fig. 1). This approach restores the ability to perform an emulation with full integrity, assuming the requisite devices are physically available. This paper describes the design constraints, implementation decisions and architecture of our Cyber-physical Emulation Test Range (CPETR).

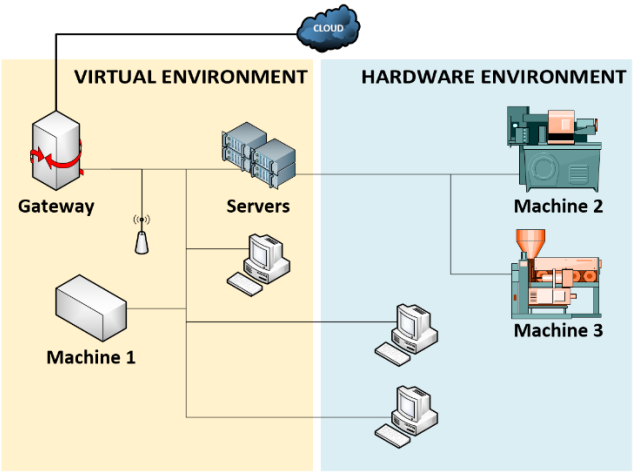

FIGURE 1. Cyber-physical test range.

As network topologies have grown increasingly complicated in recent years, there is an increased need for the ability to test a network architecture without implementing physical connections to all the planned devices. In some cases, this will be driven by the desire to verify the performance of a topology before committing to the cost of purchasing and installing the hardware. In other cases, the hardware may already be purchased but may be deemed too critical to be subject to a strenuous live test (such as a penetration test) that might leave a production system damaged or inoperable. In both cases, physical networks may be represented as emulated virtual networks running on a server, allowing for penetration testing without the cost or risk associated with a duplicate set of hardware [3].

These virtualization strategies can be broken down into two key approaches: simulations and emulations. Simulations are

used to model large, complex networks with low fidelity nodes on the network. These nodes are capable of simulating the traffic between nodes but lack a full implementation of the system that they represent (OS, memory and network capabilities). Emulations utilize high fidelity representations of actual machines and devices for the nodes. These emulated nodes provide an authentic picture of the capabilities and weaknesses of an individual node but come with an added computational cost. The Global Environment for Network Innovations (GENI) is an example of an emulation package that is capable of virtualizing up to 50 nodes with OSI Layer 2 connectivity, deploying custom software and operating systems on these nodes, and managing Layer 3 and above protocols [4]–[6]. Another example is the Common Open Research Emulator (CORE), which also operates at Layer 3 and above, and is limited to be able to emulate only 10's of nodes [7]. Other widely-used network emulators include Emulab, OFELIA, V-network, GNS3 and NEmu [8]–[10].

As standalone applications, however, GENI and CORE come with a significant limitation in that they can only generate a network of virtual machines and are unable to connect actual physical hardware. While the desire to add physical devices to a virtualized network may seem to run counter to the goals of virtualization listed above, there are significant instances where it would be advantageous, if not required, to have a hybrid network comprised of both physical and virtual assets. As networks grow to include more varied components, effective emulation of specific hardware may not be economical or even feasible. This is particularly true as networks expand to include devices outside of traditional networking such as Internet of Things (IoT) devices, programmable automation controllers (PACs) and programmable logic controllers (PLCs), or manufacturing equipment such as computer numerically controlled (CNC) mills and lathes. An analyst setting up a test network with these physical elements might find it easier to connect the actual device to the network of virtual nodes instead of attempting to virtualize an obscure piece of hardware. A package such as NEmu can interface with physical hardware, but requires that hardware be running Linux, a severe limitation for a robust solution.

II. BACKGROUND

The focus of these previously discussed emulators is completely virtual in nature; they allow for a variety of frameworks and operate at different levels of node fidelity. Our current research, which has a network-security focus, required an emulation testbed that allowed for very high-fidelity nodes of various flavors (e.g. servers, managed switches, MS Windows PCs, Linux PCs, and industrial equipment). In addition to these nodes, which are likely exact clones of real physical machines, we had a requirement to be able to include actual physical nodes within the virtual network.

Test-beds have been created in the past to allow for such hardware-in-the-loop (HIL) emulation, but these tend to be built and customized around specific hardware and specific applications [11]–[13]. The current research developed a network emulation architecture (Fig. 2) that allows for multiple physical nodes to be included on the virtually emulated network, addressing them only at the ethernet level. This allows for a completely flexible HIL emulation that requires no additional development; any ethernet device plugged into a designated set of physical switch ports is seamlessly included in the virtual network's Level 2 space. The degree of integrity of the emulated network is such that a penetration test run inside the virtual network sees the physical devices as integral to the virtual network under test.

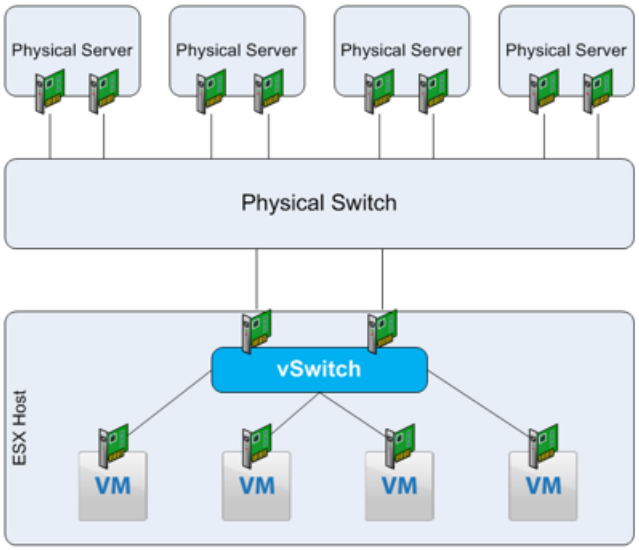

FIGURE 2. Cyber-physical connection using a vSwitch.

The purpose of this research as a new method for network emulation is to have a greater compatibility by accommodating all networked devices that might be found in a manufacturing environment, including any physical device that cannot be cloned virtually. This requirement is critical to ensure that test coverage reaches the entire hardware and software spectrum. Most HIL emulators are focused on a specific hardware or software [14], [15]. We need to emulate manufacturing networks without the limitations of only being able to test a limited set of devices and software programs.

III. DESIGN APPROACH

One of the design requirements of the emulation test range was to leverage mature, commercially available and/or open source technologies. As such, we decided to develop the CPETR on the customer-approved virtualization platform, VMware. The goal was to remove as many questions of orchestration as possible, such as concerns about allocating physical resources, bridging requirements that arise when connecting a new physical device, and non-default configuration requirements.

Our first design challenge was how to emulate a network such that the VMware infrastructure was invisible to a penetration tester. The CPETR had to appear to the tester

in such a way to make it seem as though they were on a physical network with no visibility of the management architecture. To achieve this illusion, we designed the range on OSI Layer 2 constructs using Layer 2 switches [16]. While some existing cyber ranges use Layer 1 (crossbar) switches, they are far more expensive and lag behind current Ethernet standards, typically only supporting a 1-gigabit backbone for emulation traffic. We felt that introducing these limitations was not a sufficient tradeoff for the independence from Layer 2 protocols that crossbar switches would bring. Additionally, Layer 1 is already provided by the virtual hardware in virtual machines (E1000, E1000e, VMXNET2, VMXNET3, etc) [17] or by the physical equipment that would be connected to one of the physical ports for the system.

Finally, the Ethernet standard was developed to emulate the shared cable of legacy bus or ring network topologies. As such, all hosts connected to the same Layer 2 segment can read all traffic on that segment. For instance, all hosts connected to an unmanaged network switch can typically read all of the traffic flowing through that switch. Therefore, this promiscuous nature of Layer 2 is needed in a high-fidelity network emulation, which is also critically important when performing penetration tests where a malicious host will try to capture communications of other hosts [18], [19].

We built the Layer 2 infrastructure of the CPETR on VMware's vSphere Distributed Switch (VDS). Internally, VDS uses its own set of frame forwarding techniques such that Layer 2 segments can extend to multiple VM hosts, hence the virtual switch is ''distributed'' across multiple hosts [20]. A VDS can be further broken down into Portgroups, which is a group of ports on a virtual switch (vSwitch). When multiple virtual nodes occupy the same Layer 2 segment, they are added to the same Portgroup, which has a unique VLAN ID.

It is important to note that while VDS is built on OVS (Open vSwitch), VMware does not give users full control [21], [22]. As such, there are design limitations to consider:

1) Two virtual machine network interfaces could not be bridged directly to one another; instead, a user would need to configure a generic switch, or two-port Portgroup, between the two interfaces.
2) VDS ports could not be bridged directly to one another. This is important for our purposes because it would prevent a user seeking to model a network that contains two unmanaged network switches linked via an Ethernet cable from linking two generic switches in our range. The user would have to combine the two switches into a single Layer 2 segment in the network model or would have to treat VLANs as Layer 2 segments.
3) VDS in VMware forges gratuitous ARP messages on behalf of virtual machines by default which would create anomalous network behavior in the emulation that does not match what would happen in the physical network being emulated. As such, we disabled this option on all VDS components involved in emulations.
4) VDS blocks frames with spoofed MAC addresses by default. While this is a good practice for security in typical VMware use cases, we do not want this in emulations where an attacker would normally be able to utilize MAC spoofing. As such, we disabled this option on all VDS components involved in emulations.
5) VDS prevents virtual machines from putting their network interfaces in promiscuous mode by default. Promiscuous mode is used to snoop on all traffic traversing that Layer 2 segment. While preventing promiscuous mode is a good practice for security in typical VMware use cases, we do not want this in emulations where an attacker would normally be able to place network interfaces on compromised hosts in promiscuous mode. As such, we disabled this option on all VDS components involved in emulations.

While designing the range, we also discovered that directed and undirected network graphs left ambiguity when it came to understanding precisely how nodes connect to one another. Interface numbers had to be respected if we were to remove this ambiguity. For example, if one node acted as a firewall, it mattered which interface was treated as the secure network (LAN side). Internally, most firewalls and routers are configured to associate interface numbers with zones, rules, and so forth. Therefore, we designed the CPETR to allow users to specify precisely which interfaces on each node connected to a particular Layer 2 segment, as shown in Fig. 3. In addition, the VDS Portgroups were created with only enough ports for the nodes connected to it. If a physical node was connected to that segment, there would be a specific port on the Portgroup reserved for it.

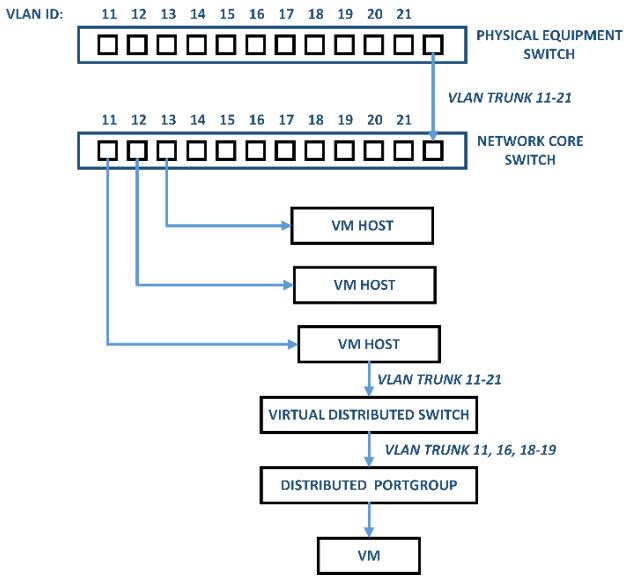

FIGURE 3. Infrastructure (underlay) configuration.

We used Virtual Local Area Networks (VLAN) to create a separation of traffic between Portgroups on the same VDS in our range. VLANs allow us to preserve logical separation

between the various Layer 2 segments within the same VDS, even though they are physically connected to two or more Local Area Networks (LAN) [23]. This is essential for our purposes, as our range needs to connect physical devices in addition to virtual devices and requires them to behave as though they are all on the same physical network.

The use of a VLAN approach creates an upper-bound of 4094 IDs for any network, due to the 12-bit header. We assign one VLAN ID to each network interface of a physical node, and one VLAN ID to each generic switch used to make the interconnects. The ID quantity constraint does not affect our range because our required node capacity is 350, and our goal capacity is 1000. It is worth noting that our architecture can extend past the 4094 limit by using a Virtual Extensible LAN (VXLAN) approach. However, given that our virtualization infrastructure was VMware, doing so would have required us to use VMware's NSX-V; this would have increased the infrastructure cost considerably for implementers of our range software, whereas the VLAN-based approach imposes no cost increase. The VLAN-approach also imposed a second limitation that would not be imposed by a VXLAN approach: nodes in an emulation cannot do their own 802.11Q VLAN tagging (VLAN trunking). However, we were able to work around this limitation by decomposing the problem in terms of Layer 2 segments.

Our design uses one or more managed switches connected to the emulation backbone to accommodate our customer's primary requirement of connecting physical devices directly into the CPETR. For every physical device network interface used in an emulation, the range architecture creates a dedicated VDS Portgroup with a VLAN ID matching the VLAN ID that the physical device network interface is connected to on the physical device switch. Using VLANs guarantees separation of traffic between physical devices in an emulation unless those physical devices belong to the same Layer 2 segment in the virtual realm (i.e., they belong to the same VDS Portgroup).

This single switch or collection of switches would have to have enough interfaces to connect every network interface of all physical devices that could be used in any one emulation, plus enough uplink ports to connect these switches to the emulation backbone. Each physical device switch would be configured such that each port had a unique VLAN ID. As the VLAN IDs used for physical devices and the VLAN IDs used to separate VDS Portgroups share a VLAN space, they could not overlap.

This approach ensured that every frame sent out of a network interface belonged to a physical device node of an emulation and would be forwarded to the appropriate VDS Portgroup port of the OSI Layer 2 segment specified as if that physical device were plugged directly into that virtual port (Fig. 4). Each frame sent to that physical device interface would be sent out of that VDS Portgroup port and would be forwarded until the frame reached the network interface of that physical device. These frames would arrive unchanged

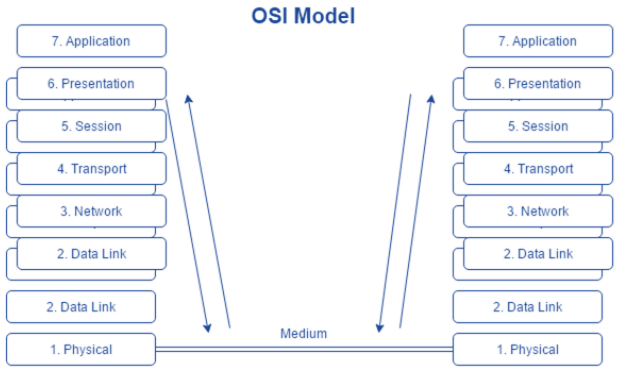

FIGURE 4. Overlay networking on OSI model.

at their destination and therefore these virtual links would be indistinguishable to the nodes on the link from any other network links.

As mentioned earlier, there is a VDS Portgroup port reserved for every physical device network interface on the Layer 2 segment that the network interface belongs to in an emulation. There is also a second VDS Portgroup with a switch port connected to the physical device's network interface. It was necessary to bridge these two VDS Portgroups. To do so, we used a lightweight virtual machine to act as a Layer 2 bridge. This virtual machine had two network interfaces and simply forwarded any frame received on one interface to its other interface, unchanged. This virtual machine is undetectable to other nodes on that Layer 2 segment as it has no IP address and never sends a frame exposing its own MAC addresses. By bridging the virtual network adapter inside this VM with the physical port on the hardware switch, VMware vSphere provides built-in time synchronization for the packets passing between the two.

IV. IMPLEMENTATION

We implemented our range on VMware ESXi hosts with two physically separate networks: one for management/ orchestration traffic, and another for emulation traffic. Each ESXi server (we initially used two, but it is scalable) had at least one network interface connected to the management network and another network interface connected to the emulation backbone. The emulation backbone managed switch was configured to act as VLAN trunk ports to tag all VLANs on all ports. This guaranteed that nodes inside of an emulation could not detect the presence of the CPETR infrastructure so long as host/guest separation was not violated. Each ESXi host network interface connected to the emulation backbone was also a member of the VDS (uplink ports). Our implementation of the bridge virtual machine to accommodate physical devices was built on a lightweight version of Debian Linux.

One potential limitation of our design to accommodate physical devices is link negotiation. What happens when someone wants to connect a physical device that can only communicate in half-duplex mode? To ensure no

duplex-mismatch issues arise, the system would have to ensure that link negotiation messages are passed through the system appropriately such that it will never create a situation where one node on the virtual link believes the link is full-duplex and the other node thinks the link is half-duplex. This might not be possible to guarantee in a system based on VMware's VDS since someone would not have full control over the underlying OVS. Furthermore, this problem would not exist if the network virtualization system is built on VXLAN rather than VLAN.

Multiple penetration tests have been performed on a variety of emulated networks built using CPETR. Typically, a Kali Linux workstation is connected to the network. Performing tests such as using a Metasploit framework to find open ports on servers or workstations within the network have performed as expected.

Performance data on the ability of CPETR to build, start and teardown a heterogeneous network have been collected. Typical results for a 350-node network are shown in Fig. 5. The network builds in just over 10 minutes, all nodes are started in less than six minutes and complete teardown takes approximately 7 minutes.

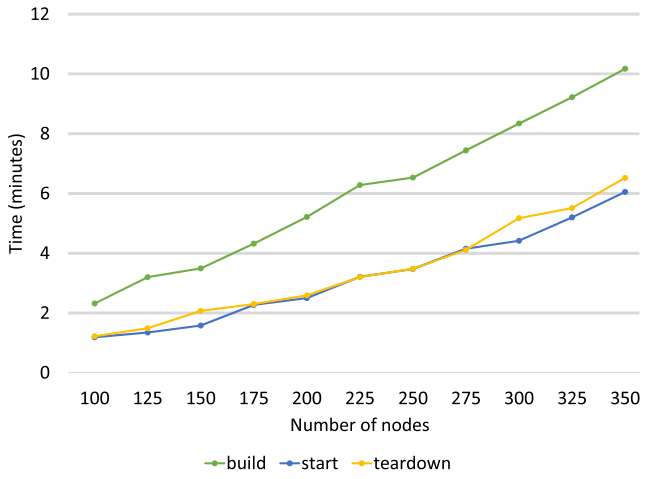

FIGURE 5. CPETR performance data.

V. RESULTS

We created a test plan to verify that CPETR was able to emulate a physical network using some HIL devices without requiring any custom configuration. The testing consisted of creating a closed network meant to simulate a small manufacturing firm; it was composed of three subnets with a total of twenty-five physical devices. The first subnet, simulating the factory floor, contained nine Microsoft Windows 7 machines and four Automation Direct CLICK Programmable Logic Controllers (PLCs); each of these devices was directly connected into an unmanaged switch named *Main Switch*; a router was also connected to this switch acting primarily as a DCHP server for the entire network. The second subnet, simulating an IT department, was composed of three Kali

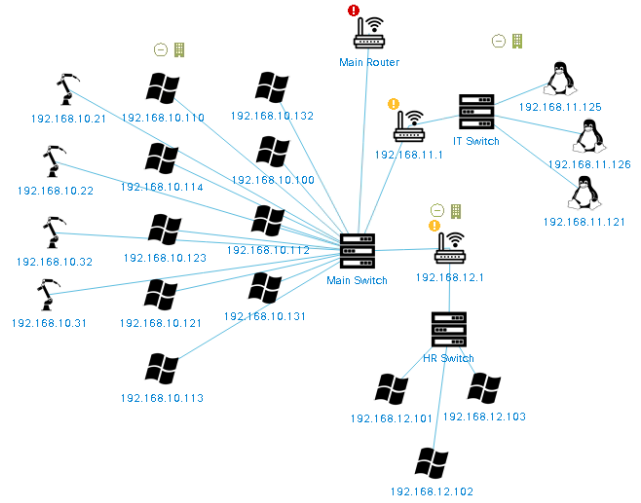

FIGURE 6. Test network topology.

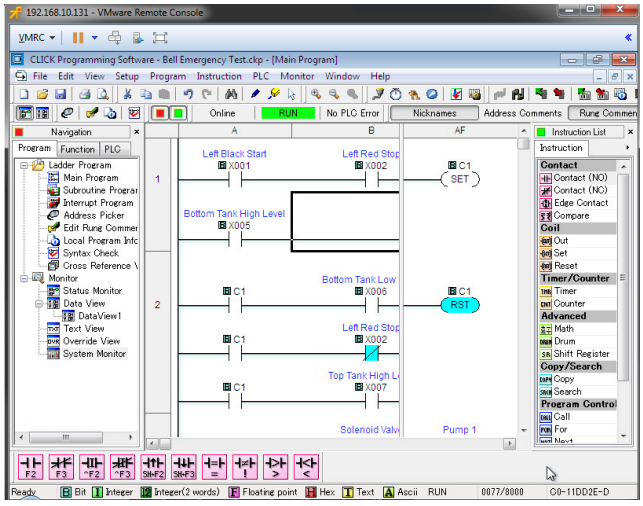

FIGURE 7. PLC test running in interactive mode.

machines connected to an unmanaged switch which was connected to a firewall, and in turn to the Main Switch. The third subnet, meant to simulate an HR department, was composed of three Windows 7 machines connected to an unmanaged switch then to a firewall, and finally to the Main Switch. We then used an Nmap tool to scan the network to inventory every node on the network, giving us a complete network audit of the physical network. We then cloned all the nodes that were virtualizable (i.e. the Windows machines, the Kali machines, and the firewalls); devices that could not be cloned (i.e. the PLCs) were physically plugged into the CPETR physical switch (an isolated Dell X1052). We emulated the exact physical network in CPETR including the actual PLCs devices used in the physical network topology, as shown in Fig. 6. In order to validate the functionality of the network, we repeated the Nmap scans and compared

the results. We then performed a validation using one of the Windows 7 VMs to read a project file from the CLICK PLC hardware (that had previously been downloaded into the PLC). After the CLICK PLC software on the VM read the project, we then modified the program and wrote it back to the PLC for running a bell emergency test. Fig. 7 shows the CLICK PLC software with the results being read directly from the hardware after the modification of the project file.

VI. CONCLUSION

An architecture was successfully designed and built using a VMware foundation to allow a true cyber-physical emulation test bed to be created. The Cyber-physical Emulation Test Range allows portions of a network to be emulated while simultaneously allowing physical nodes to seamlessly exist on the same network in real time. This allows for the recreation of a cloned network, even when some of the nodes are not able to be virtualized. Alternatively, this allows for thorough testing of a physical device on a network in advance of connecting it to the real network.

A significant benefit of being able to truly emulate an entire network, including non-virtualizable elements, is the ability to perform thorough penetration testing without the risks inherent in doing so on a live or production network. Because the emulated network acts as the original network on all layers, the full variety of penetration tools can be used in the test-bed environment.

By harnessing the authoritative toolsets provided by VMware and customizing them in ways to allow the emulated network to be isolated unto itself, a powerful platform was able to be created. Since the backbone is VMware, CPETR can be continually updated and benefit from any patches or updates that are released for VMware. This allows for it to remain current and significant on an ongoing basis, with minimal maintenance.

VII. FUTURE WORK

Additional work is planned to increase the efficiency of creating the network with the goal to increase the number of nodes that can be emulated to one thousand; the immediate limitation is sufficient available memory, hard drive space, and processor cores, all of which need to be expanded in order to accommodate the additional VMs. Also, a customized front-end graphical user interface is planned for the CPETR that will allow networks to be built on the range in a drag-and-drop fashion once a library of nodes is created. Lastly, a network scanning tool is under development that will allow a network to be scanned and its emulated clone automatically built in CPETR.

REFERENCES

[1] T.-S. Chou, S. Baker, and M. Vega-Herrera, "A comparison of network simulation and emulation virtualization tools," in *Proc. ASEE Annu. Conf. Expo.*, 2016, pp. 1–9.

[2] Y. A. Shichkina, M. S. Kupriyanov, and S. O. Moldachev, "Application of Docker Swarm cluster for testing programs, developed for system of devices within paradigm of Internet of Things," *J. Phys., Conf. Ser.*, vol. 1015, p. 032129, May 2018.

[3] A. Brokalakis, N. Tampouratzis, A. Nikitakis, S. Andrianakis, I. Papaefstathiou, and A. Dollas, "An open-source extendable, highly-accurate and security aware CPS simulator," in *Proc. 13th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, Jun. 2017, pp. 81–88.

[4] M. Berman *et al.*, "GENI: A federated testbed for innovative network experiments," *Comput. Netw.*, vol. 61, pp. 5–23, Mar. 2014.

[5] N. Van Vorst, M. Erazo, and J. Liu, "PrimoGENI: Integrating real-time network simulation and emulation in GENI," in *Proc. IEEE Workshop Princ. Adv. Distrib. Simulation*, Jun. 2011, pp. 1–9.

[6] A. R. Roy, S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "Emulating an infrastructure with EASE," in *Proc. 12th Int. Conf. Netw. Service Manage. (CNSM)*, Oct./Nov. 2016, pp. 167–173.

[7] J. Ahrenholz, "Comparison of CORE network emulation platforms," in *Proc. Milcom Mil. Commun. Conf.*, Oct./Nov. 2010, pp. 166–171.

[8] P.-W. Tsai, F. Piccialli, C.-W. Tsai, M.-Y. Luo, and C.-S. Yang, "Control frameworks in network emulation testbeds: A survey," *J. Comput. Sci.*, vol. 22, pp. 148–161, Sep. 2017.

[9] M. A. Ahmad, S. Woodhead, and D. Gan, "The V-network testbed for malware analysis," in *Proc. Int. Conf. Adv. Commun. Control Comput. Technol. (ICACCCT)*, May 2016, pp. 629–635.

[10] H. Gao, Y. Peng, Z. Dai, and H. Li, "Techniques and research trends of network testbed," in *Proc. 10th Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.*, Aug. 2014, pp. 537–541.

[11] K. Christakou, M. Pignati, R. Rudnik, S. Sarri, J.-Y. Le Boudec, and M. Paolone, "Hardware-in-the-loop validation of the grid explicit congestion notification mechanism for primary voltage control in active distribution networks," in *Proc. Power Syst. Comput. Conf. (PSCC)*, Jun. 2016, pp. 1–7.

[12] F. Han *et al.*, "A real-time virtual simulation environment for advanced driver assistance system development," SAE Tech. Paper 2014-01-0194, 2014.

[13] M. Iacob, G.-D. Andreescu, R. Antal, and A.-M. Dan, "Multivariable adaptive control with hardware-in-the-loop for a drum-type boiler-turbine system," in *Proc. 19th Medit. Conf. Control Autom. (MED)*, Jun. 2011, pp. 893–903.

[14] J. H. Jeon *et al.*, "Development of hardware in-the-loop simulation system for testing operation and control functions of microgrid," *IEEE Trans. Power. Electron.*, vol. 25, no. 12, pp. 2919–2929, Dec. 2010.

[15] B. Palmintier, B. Lundstrom, S. Chakraborty, T. Williams, K. Schneider, and D. Chassin, "A power hardware-in-the-loop platform with remote distribution circuit cosimulation," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2236–2245, Apr. 2015.

[16] Y. Li, D. Li, W. Cui, and R. Zhang, "Research based on OSI model," in *Proc. IEEE 3rd Int. Conf. Commun. Softw. Netw.*, May 2011, pp. 554–557.

[17] S. Zhou, "Virtual networking," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 4, pp. 80–85, 2010.

[18] T. Solomon, A. M. Zungeru, and R. Selvaraj, "Network traffic monitoring in an industrial environment," in *Proc. 3rd Int. Conf. Electr., Electron., Comput. Eng. Appl. (EECEA)*, Apr. 2016, pp. 133–139.

[19] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. Bhattacharyya, and J. K. Kalita, "Network attacks: Taxonomy, tools and systems," *J. Netw. Comput. Appl.*, vol. 40, pp. 307–324, Apr. 2014.

[20] Y. Luo, E. Murray, and T. L. Ficarra, "Accelerated virtual switching with programmable NICs for scalable data center networking," in *Proc. 2nd ACM SIGCOMM Workshop Virtualized Infrastruct. Syst. Archit. (VISA)*, vol. 10, 2010, pp. 65–72.

[21] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker, "Extending networking into the virtualization layer," in *Proc. HotNets*, 2009, pp. 1–6.

[22] B. Pfaff *et al.*, "The design and implementation of open vSwitch," in *Proc. 12th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2015, pp. 117–130.

[23] A. Luntovskyy and J. Spillner, "Cloud Computing, Virtualisation, Storage and Networking," in *Architectural Transformations in Network Services and Distributed Systems*. Wiesbaden, Germany: Springer, 2017, pp. 77–133.

AIDAN F. BROWNE received the B.S. degree in physics and mathematics from Vanderbilt University, Nashville, TN, USA, in 2008, and the M.S. degree in biological engineering, the M.S. degree in electrical engineering, and the Ph.D. degree in biomedical engineering from the University of Connecticut, Storrs, CT, USA, in 1995, 1997 and 1998, respectively.

From 1998 to 2001, he was an Electrical Systems Engineer with Hamilton Sundstrand Space Systems International. From 2001 to 2004, he was an Electrical Systems Engineer with Hamilton Sundstrand Control Systems. From 2004 to 2010, he was a Lead Systems Engineer with General Dynamics Armament and Technical Products. He is currently an Assistant Professor with the Department of Engineering Technology and Construction Management, The University of North Carolina at Charlotte. His research interests include robotics, mechatronics, instrumentation and sensing.

Dr. Browne currently serves as the Chair of the IEEE Charlotte Section. He holds memberships in the American Society of Engineering Education and the Vibration Institute.

STACEY L. WATSON was born in Welland, ON, Canada, in 1968. She received the B.A. degree in economics from York University, Toronto, ON, Canada, in 1989, the B.Ed. degree in intermediate/senior english and history education from Brock University, St. Catharines, ON, Canada, in 1999, the M.S. degree in applied computer science from Columbus State University, Columbus, GA, USA, in 2013. She is currently pursuing the Ph.D. degree in software and information systems with the University of North Carolina (UNC) at Charlotte, Charlotte, NC, USA.

From 2012 to 2014, she was a Security Analyst with Total Systems, Inc., from 2014 to 2015, she was a Security Engineer at Hewlett-Packard, and from 2015 to 2016, she was a Senior Security Engineer with the Intercontinental Hotels Group. Since 2016, she has been a Lecturer with the Department of Software and Information Systems, UNC at Charlotte. Her research interests include cybersecurity, usable security, and computer science education.

Ms. Watson is a member of the Association for Computing Machinery and InfraGard.

WESLEY B. WILLIAMS received the B.S. degree from North Carolina State University in 2000, the M.S. degree from the Georgia Institute of Technology in 2005, and the Ph.D. degree from The University of North Carolina at Charlotte in 2009, all in mechanical engineering.

From 2000 to 2003, he was a Mechanical Design Engineer with GlaxoSmithKline, designing automation and instrumentation solutions for pharmaceutical researchers. Upon completing his Ph.D., he was a Post-Doctoral Researcher with the NASA Center for Aviation Safety, North Carolina A&T State University. In 2011, he joined the Department of Engineering Technology and Construction Management, The University of North Carolina at Charlotte, as an Assistant Professor and was promoted to an Associate Professor in 2017. He is licensed as a professional engineer in North Carolina. His research interests include magnetic gearing, industrial automation, and additive manufacturing.