Emerging Network and Architecture Technology Enhancements to Support Future Training Environments

Bruce Caulkins, Ph.D., Brian Goldiez, Ph.D.,
Paul Wiegand, Ph.D., Glenn Martin, Ph.D.
Institute for Simulation & Training (IST)
University of Central Florida (UCF)
Orlando, Florida
bcaulkin@ist.ucf.edu, bgoldiez@ist.ucf.edu,
wiegand@ist.ucf.edu, martin@ist.ucf.edu

Paul Dumanoir,
Tom Torres
U.S. Army PEO STRI
Orlando, Florida
paul.h.dumanoir.civ@mail.mil,
thomas.e.torres2.civ@mail.mil

ABSTRACT

The Operational Environment (OE) has become increasingly complex, with challenging human factors, exponential proliferation of technology, and an increasingly determined, adaptive threat. Training Army Soldiers, leaders and units in a complex world requires modernized, integrated, realistic, and adaptive training capabilities. The Army must leverage emerging technologies to transform the way it develops and delivers training to enable agile and adaptive Soldiers, leaders and versatile units. It must provide a consistent, persistent ability to train at the point of need (PON) for both current and future operations as part of a Joint, Inter-organizational, and Multinational (JIM) force. The training venues must allow the Army to train as it fights, using its wartime systems on its operational networks and all training environments must replicate the OE to the greatest extent possible.

To address this reality, the U.S. Army Program Executive Office for Simulation, Training and Instrumentation (PEO STRI) and the University of Central Florida (UCF), Institute for Simulation and Training (IST) are conducting research to create new capabilities that support both operations and training while enabling software application migration to Army enterprise data centers and cloud environments. This research pivots on Army directives that focus on modernizing information technology systems and applications. To achieve the distributed nature of this vision, technical enhancements to the underlying Army Enterprise Network (AEN) must be made in the next few years.

This paper investigates potential gaps in simulation enterprise network architectures and describes research results in three major technical areas that address these gaps and will benefit future simulation and network architectures. The research topics include technologies that: (1) efficiently delivers simulations from cloud-like environments using Software Defined Networks (SDNs); (2) facilitates individual and collective home station or field-based training through the use of thin clients; and (3) optimizes computational resources through load-balancing techniques and processes.

ABOUT THE AUTHORS

Bruce Caulkins is a Research Assistant Professor at the Institute for Simulation and Training at UCF, focusing on cybersecurity-related research and instruction. He is also the Program Director for UCF's Modeling and Simulation of Behavioral Cybersecurity graduate certificate program. He is a retired Army Colonel with over 28 years of experience in tactical, operational, and strategic cyberspace operations. He earned his Ph.D. in Modeling and Simulation at UCF in 2005, focusing on anomaly detection within intrusion-detection systems.

Paul Dumanoir is the Chief Engineer for the United States Army Product Manager for Warrior Training Integration (PdM WTI) under the Project Manager for Integrated Training Environment (PM ITE) at PEO STRI. He is currently the PM ITE Risk Reduction Test Bed project manager and has 26+ years of experience working in Army and DoD simulation and training programs as Product Manager, Project Director, and Systems / Software Engineer. His current

interests include component-based product-line engineering, enterprise architectures, and system of system integration and interoperability. He earned his B.S. in Electrical Engineering from the University of South Alabama in 1987 and his M.S. in Computer Systems from the University of Central Florida in 1991.

Brian Goldiez is a Research Associate Professor and Deputy Director at the University of Central Florida's Institute for Simulation and Training. He has over 40 years of experience in the Modeling and Simulation field in government, industry, and academia. His work has covered live, virtual, constructive and gaming with particular emphasis on human and simulation performance measurement and optimization. Recent research has involved interoperability data structures, training effectiveness measurement and evaluations, and new methods to generate computer generated imagery. His current efforts involve several programs in medical simulation, cloud based delivery of simulation, high performance computing, and Unmanned Aerial Systems. Goldiez has a Ph.D. from the University of Central Florida in Modeling and Simulation.

R. Paul Wiegand is a Research Assistant Professor at the University of Central Florida's Institute for Simulation and Training. Dr. Wiegand's research interests include machine learning, advanced heuristic optimization methods, multiagent simulation, and high performance computing (HPC). He directs the Natural Computation and Coadaptive Systems laboratory and co-manages compute and network resources at the Advanced Research Computing Center, which include UCF's SDN-enabled research network.

Glenn Martin is a Research Assistant Professor at the University of Central Florida. He leads the Interactive Realities Laboratory at the University of Central Florida's Institute for Simulation and Training. He earned a Ph.D. in Modeling & Simulation in 2012 from the University of Central Florida, specializing in adaptive training through automated scenario generation. He pursues research in adaptive and intelligent training, game-based learning, multi-modal simulation, competitive learning and interactive high performance computing.

Tom Torres is the Systems Engineer for the Live, Virtual, and Constructive Integrated Architecture (LVC-IA) Program of Record for the United States Army Product Manager for Warrior Training Integration (PdM WTI) under the Project Manager for Integrated Training Environment (PM ITE) at PEO STRI. He is currently the LVC-IA New Equipment Training Lead and has 8+ years of experience working in Army and DoD simulation and training programs as Systems / Software Engineer.

Emerging Network and Architecture Technology Enhancements to Support Future Training Environments

Bruce Caulkins, Ph.D., Brian Goldiez, Ph.D.,
Paul Wiegand, Ph.D., Glenn Martin, Ph.D.
Institute for Simulation & Training (IST)
University of Central Florida (UCF)
Orlando, Florida
bcaulkin@ist.ucf.edu, bgoldiez@ist.ucf.edu,
wiegand@ist.ucf.edu, martin@ist.ucf.edu

Paul Dumanoir,
Tom Torres
U.S. Army PEO STRI
Orlando, Florida
paul.h.dumanoir.civ@mail.mil,
thomas.e.torres2.civ@mail.mil

BACKGROUND

The Operational Environment (OE) has become increasingly complex, with challenging human factors, exponential proliferation of technology, and an increasingly determined, adaptive threat. Training Army Soldiers, leaders and units in a complex world requires modernized, integrated, realistic, and adaptive training capabilities. The Army must leverage emerging technologies to transform the way it develops and delivers training to enable agile and adaptive Soldiers, leaders and versatile units. It must provide a consistent, persistent ability to train at the point of need (PON) for both current and future operations as part of a Joint, Inter-organizational, and Multinational (JIM) force. The training venues must allow the Army to train as it fights, using its wartime systems on its operational networks and all training environments must replicate the OE to the greatest extent possible.

To address this reality, the U.S. Army Program Executive Office for Simulation, Training and Instrumentation (PEO STRI) and the University of Central Florida (UCF), Institute for Simulation and Training (IST) are conducting research to create new capabilities that support both operations and training while enabling software application migration to Army enterprise data centers and cloud environments. This research pivots on Army directives that focus on rationalizing and modernizing information technology systems and applications.

To achieve the distributed nature of this vision, technical enhancements to the underlying Army Enterprise Network (AEN) must be made in the next few years. In particular, this research focuses on technical enhancements to improve the Army's Integrated Training Environment (ITE) system of systems which uses the AEN to bring together a persistent set of Live, Virtual, Constructive, and Gaming (LVC-G) training capabilities and provide realistic training at Mission Training Centers (MTCs).

This paper investigates three current gaps in simulation enterprise network architectures and describes research results in three major technical areas that will benefit future simulation and network architectures. The gaps being addressed span the server, client, and network connectivity research areas. The specific research topics include technologies that will: (1) efficiently deliver simulations from cloud-like environments using Software Defined Networking (SDN); (2) facilitate individual and collective home station or field-based training through the use of thin clients; and (3) optimize computational resources through load-balancing techniques and processes. The paper describes how specific new technologies and tools, in these three areas, can be leveraged to improve the ability to deploy more configurable, optimizable, and flexible training simulations which will allow the ITE more versatility in terms of networking and content delivery.

THE THREE GAPS AND RELATED RESEARCH ACTIVITY

Software Defined Networking (SDN)

For the future ITE to function efficiently, the Army will need to be capable of adaptive provisioning and optimization of the network mechanisms that support those activities. Such a capability will need to address some fundamental issues, such as bandwidth and latency limitations both within and between facilities, competition for bandwidth between both ITE and non-ITE network activities, and the high variance in bandwidth utilization among simulation tools. Ideally, a network solution capable of addressing these concerns will be configurable and adaptable to a variety of different contexts, depending on the tools used by and the needs of a given set of training exercises. In particular, what is needed is an agile network that uses existing Army infrastructure and can be reconfigured rapidly and centrally in a cost-effective way.

PEO STRI & IST examined the use of SDN to meet these needs. SDN provides programmable and adaptive mechanisms to support network virtualization and functional separation, giving both network administrators, as well as software-based provisioning tools the ability to provide services where they are desired in a network, without regard to specific connected or connecting devices (Hu 2014, Ichelson 2016). As depicted in Figure 1, SDN technologies can be separated into three layers: the *SDN communication protocol* that resides at the switch level, the *SDN controller software* that runs server(s) within the network, and *SDN application software* that can run from various computers within the network. This layering provides a means to reconfigure the network *at the control plane level* through software.

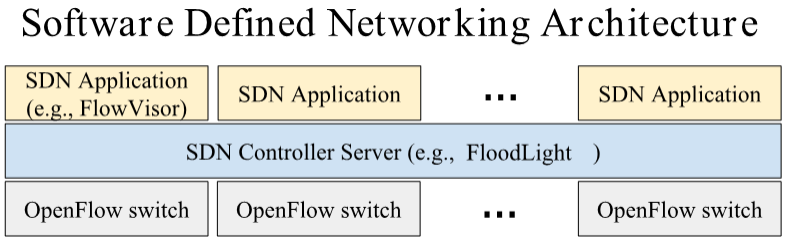

Figure 1. General Architecture Diagram for SDN

For our research, we used a widely adopted open communication standard for SDN, which is *OpenFlow* (v1.3). We used another commonly adopted tool, *FloodLight* to serve as our SDN controller. FloodLight supports multiple versions of OpenFlow concurrently, has a web-based GUI, is known to perform and scale well, contains network topology discovery mechanisms, exposes a convenient Python-based API, and has strong community and documentation support.

To address bandwidth and latency issues, we explored mechanisms to implement adaptive control plane flow management within the switches to produce separate logical network *slices* within existing hardware. This means that network flows between machines in one logical space do not compete with network flows within another, even if those flows make use of the same physical infrastructure. This provides a means of assigning machines and portions of switches to multiple networks on the fly, via software or from archived configurations without having to reconfigure multiple scripts and Network Interface Cards (NICs) within the network. In this way, ITE traffic and non-ITE traffic can be isolated from one another and will not compete.

To do this, we made use of the off-the-shelf open tool, *FlowVisor as shown in Figure 2*. FlowVisor provides switch virtualization such that the same hardware forwarding plane is used concurrently by different logical networks, each with its own forwarding logic. Authenticated users or applications can slice physical infrastructure into different logical networks such that data is isolated and protected and guarantees quality of service. For example, machine A on one switch can be placed in the same logical network as machines C and B on another switch, while machine D on that same switch is in a completely different logical network—and these assignments can be reconfigured programmatically.

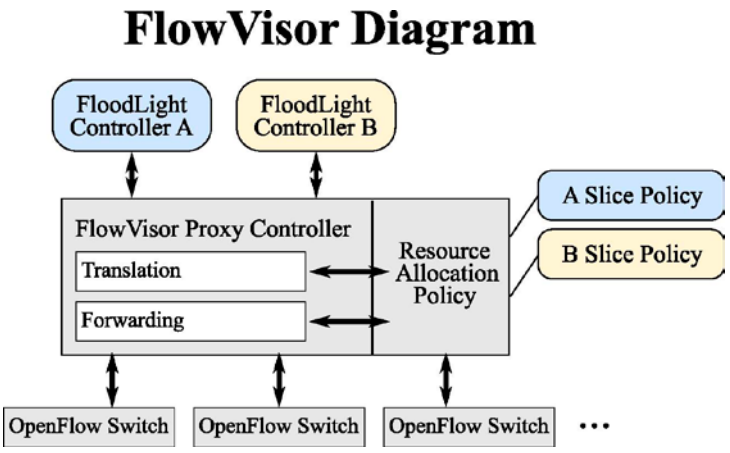

Figure 2. Sketch of One Example for Configuring Two Logical Networks Across Multiple Switches via FlowVisor

We produced the configuration scripting needed to provision such network slices on demand.

To test an SDN slicing approach, we performed the following experiments. Four nodes (A, B, C, and D) were attached to a single switch, and the bandwidth performance between two pairs of these nodes were measured (AB and DC) over a two-minute window. At the start of the second minute, node D began broadcasting traffic analogously to DIS interoperability layers in federated simulations and continued to do so for the remainder of the test. There were two experimental groups. In the first group, the switch was configured with legacy control-plane logic. In the second group, two logically separate slices were created via SDN, one slice contained nodes A and B, while the second slice contained nodes C and D. In both cases, the experiment was repeated for 10 independent trials.

Figure 3 shows the average performance results over time for these experiments. From this, we can see that slicing isolates the effects of the broadcast messages. Results at times 55s and 70s for the AB measure were compared. There is a statistically significant performance drop in the legacy network design yet no statistical evidence of a loss in performance in the SDN sliced design. Additionally, even the performance on the CD measure at time 70s (where the broadcast is originating) is statistically better under the SDN-sliced design than the legacy design. We also confirmed with the WireShark tool that no Ethernet or ARP traffic passes between nodes on different SDN slices.

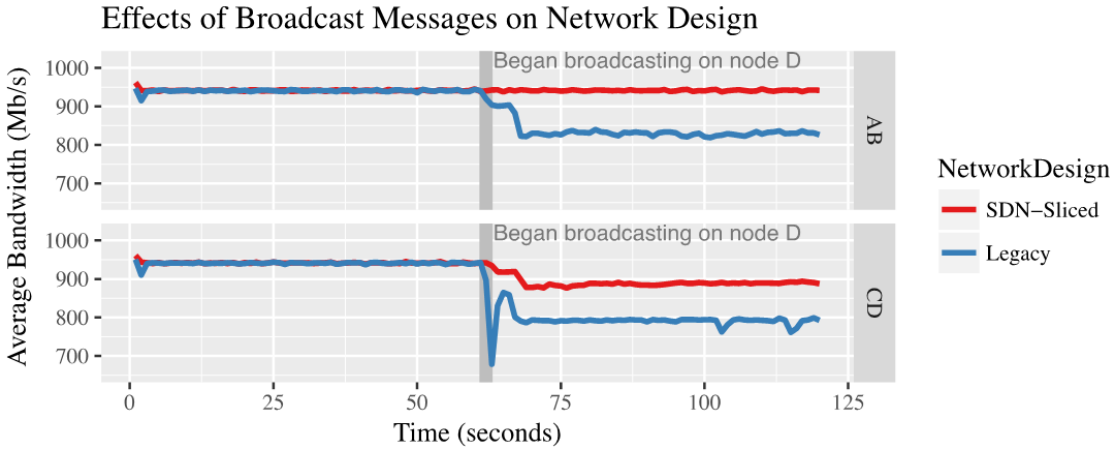

Figure 3. Effects of Broadcast Messages on Network Design

Future Work

Future work will begin using heuristic optimization methods to alter the Quality of Service (QoS) of the data flows within a given slice to optimize for nodes with different needs. For example, *after action review* nodes may require more ingress data flow, while constructive simulation engines may require more egress data flow. Flow management of this type is possible via these SDN interfaces. Also, as SDN is a new technology, continual performance enhancements have been and are expected to continue along with more rigorous standardization efforts driven by commercial enterprises.

Thin Clients

Thin clients are widely used today in many applications and domains. Many office settings use them as a cost saving measure. However, their use in virtual and constructive training has been limited despite their potential to not only reduce costs, but also to deliver training at the PON, to ease maintenance, and to provide greater opportunities for training. To drive such a system, three major components are required: a server to run the actual training application (i.e., a server to accomplish that within a virtual machine), clients that would connect to the appropriate server instance, and some kind of remote display protocol to connect the two.

The first step in our research was to identify existing gaps and requirements to be able to provide such training capabilities by thin clients. Four major gaps were identified: client hardware, network capability, server capability, and training opportunities.

(1) *Reduce the Inflated Client Hardware Footprint.* Existing training systems use either a server-client arrangement (e.g. most game-based systems) or a fully-distributed approach (e.g. DIS/HLA-based systems). Regardless of which fundamental approach is used, the client stations run relatively intensive applications in terms of computational and graphical load. This necessitates that computers (PCs) be used (purchased and maintained) that can support such applications. For example, a gaming application used for Army training, Virtual Battlespace 3 (VBS3), recommends a system with an Intel Core i5-2300 or AMD Phenom II 940 CPU, 8 GB RAM, and an Nvidia GeForce GTX 560 or AMD Radeon HD 7750 (with 1024 MB VRAM) (VBS 3, 2017). "Optimal" recommendations supporting "resource intensive scenarios" are even higher. Replicating this configuration for each trainee station at each fielded site can be costly. In addition, security and maintenance of a set of clients geographically distributed can be a challenge.

(2) *Adequate Networking Capability Required to Support Thin Clients*. To transmit the display of simulated systems to the thin client, some quantity of network bandwidth is, of course, used. For interactive virtual training, latency is also an important issue. Depending on the source, game-based training latency must be somewhere less than 60 ms (GameStream, 2017) to less than 150 ms (Tolia et al, 2006). As far as bandwidth, the requirements depend on screen resolution. Network capability will clearly be an important issue as the U.S. Army moves training to cloud-based approaches. For example, at Joint Base Lewis-McChord (JBLM), a 1 Gbps network is used, and a typical VBS3 server supports up to 52 users, and an exercise can use up to 9 servers. At Fort Bragg, a 1 Gbps network is also used internally but only a 100 Mbps network is available to the Army Network. Careful consideration of network capability will be important as it will drive potential location of cloud servers.

(3) *Servers Require Hardware Suitable for Gaming*. To drive thin clients using virtual environments for training, it's desired that the actual client software is run on a server-side computer and then a remote display protocol would be used as discussed above. Beyond an increased quantity needed, a server-side computer for thin clients must have adequate memory and graphical capability. The latter is particularly important if considering rack-mountable servers (which typically may not have an accelerated graphical capability). Support servers that include sufficient memory (RAM) and Graphical Processing Unit (GPU) capability are needed to support streaming to thin clients (multiple per server). Particularly of note, having GPU support will be required in this situation.

(4) *Provide Increased Training Opportunities.* With relatively advanced PCs required for running current training systems, this makes it difficult to expand training opportunities to other settings besides the MTCs and Sim Centers. For example, collective training or potential supplemental training within the unit (e.g. in barracks) may also be difficult. Utilization of a cloud architecture that supports thin clients can support lower-capable client PCs available at many other locations throughout the U.S. Army. If a thin client architecture is built where servers run the actual training software, which is then displayed on a thin client, this could potentially provide many additional training opportunities. While supervision and planning of these opportunities is desirable, just the option of doing it opens up possibilities. Old PCs, laptops or even a newly-purchased $50 Raspberry Pi can potentially provide the client side.

Our research focused on the first, second, and fourth of these gaps. Previous work (MITRE, 2016) explored the third gap thoroughly as they showed an ability to host a game in a virtual machine and stream it to a thin client. In their work, a general-purpose desktop display technology (XenDesktop from Citrix) was used to stream to a series of laptops and the focus was on testing the capabilities of the server. However, they did determine that six clients (running VBS3) consumed a total of 105.8 Mb/s on average. A follow-on study tested two different potential servers to host multiple clients and a server simultaneously (MITRE, 2017). Still using XenDesktop to stream, they found that the best virtual machine arrangement was one using 2 sockets, 4 cores, and the 2Q GPU profile (of the Nvidia Tesla M60 card used), and that this configuration provided 26.03 frames/second. However, their goal was to find a configuration that provided at least 30 frames/second, so they considered that goal unmet.

In our research, alternative streaming technologies were explored as we looked to answer the remaining three gaps identified above. Specifically, protocols that were built with videogames in mind were explored. While many were investigated (including GamingAnywhere, Nice DCV, and others), two quickly became the focus due to their maturity and performance characteristics. These were Nvidia GameStream and Steam In-Home Streaming. Both were studied in a formal fashion regarding network bandwidth utilization as well as in an informal fashion regarding frame rate.

To test network bandwidth, four very inexpensive thin clients were used: the Nvidia Shield (~$200), the Valve Software SteamLink (~$50), the Dragon Touch X10 Android tablet (~$100), and the Raspberry Pi 3 (~$50). Unreal Tournament 4 was used and two scenarios tested: the Weapons Training Course (a fixed path, 10-minute scenario), and Death Match (a random, combat scenario that was limited to 20 minutes). The Nvidia Shield ran its proprietary GameStream software, the SteamLink ran its proprietary In-Home Streaming software, and the Dragon Touch X10 and Raspberry Pi each ran Moonlight, which is an open-source client compatible with Nvidia's GameStream protocol. Each client's stream was hosted from a local, on-site PC.

Results for average bandwidth were as follows:

Network Bandwidth

Mbps	Weapons Training		Death Match	
	Transmit	*Receive*	*Transmit*	*Receive*
Nvidia Shield (60fps)	0.145	35.313	0.152	37.200
Valve SteamLink (30fps)	0.060	13.340	0.090	15.640
Raspberry Pi (30fps)	0.056	10.988	0.065	12.096
Android Tablet (60fps)	0.081	42.968	0.053	44.048

Figure 4. Average Network Bandwidth Results

Obviously, the bandwidth for received data was significantly larger, principally to drive the graphics displays. Of note, both the SteamLink and the Raspberry Pi had significantly reduced bandwidth utilization due to the lower frame rate of the streamed video. However, this works very well for virtual training and should allow upwards of 8 clients within a 100 Mb/s connection. The Nvidia Shield and Dragon Touch Android tablet results show the potential cost of increasing to 60 frames/sec; however, both devices support lowering down to 30 fps, and will be re-tested in future research to examine the potential bandwidth savings.

Anecdotally, all devices performed very well when driven by a native local server, and the user could not tell that a thin client is in use (frame rates of 60 frames/second or higher were consistently seen). The SteamLink was also tested when hosted by a virtual machine identical to the one used by MITRE in their study (2 sockets, 4 cores, 2Q GPU profile). While the frame rate was reduced, it still maintained rates within the 40-50 frames/second range with occasional dips to 30 frames/second.

The following conclusions are drawn:

- Thin clients are technically capable for delivering virtual training. These can include even very inexpensive clients such as consumer streaming devices, and even mobile and embedded devices (e.g. tablets and the Raspberry Pi).
- Protocols designed for streaming games seem to perform better than general-purpose streaming protocols. While XenDesktop never reached 30 frames/second in MITRE's studies, the GameStream and Steam In-Home protocols performed over that level. This should be further analyzed in a more formal study, however.
- Inexpensive thin clients have the hardware capabilities to provide additional training opportunities. If devices that cost $200 or less can support such streaming at an appropriate level, this would allow Soldiers and other personnel to train other than in scheduled visits to larger training facilities.

Future Work

Even with these encouraging results, some additional work is still appropriate. Zero clients could be studied as well as old laptops. In addition, while a single test of streaming from within a virtual machine was addressed, this should be expanded further. Future tests to be pursued will include multiple virtual machines and clients in parallel. In addition, moving the hosting of clients farther away geographically (e.g. not on-site) will be tested to explore the concept of hosting at a regional data center. This will test the streaming protocols in a setting more similar to the future potential use in cloud-based training.

Load Balancing

The load balancing effort sought to reduce the military's computer hardware footprint for simulation-based training by changing the one-to-one relationship of simulator to computer, while simultaneously automating simulation hardware provisioning and scheduling. To achieve these goals, our work is experimented with methods to optimally allocate computer resources to simulation applications a priori to and eventually dynamically during runtime. The work was developing techniques: 1) to automatically schedule and provision simulation software on remote, cloud-based servers to centralize and efficiently distribute workload across available computing resources; 2) for deploying software components on virtual machines and software containers in order to host multiple simulations per individual physical machine; and 3) to manage these virtual machines and containers by allocating and deallocating resources based on simulation CPU, memory, and network bandwidth usages during runtime.

Reducing the footprint

One of the goals of the load balancing efforts was the reduction of computer hardware used to support military training exercises. For example, typically each simulator has its own, dedicated physical computer system. There are some benefits to this current approach in the acquisition process, but some drawbacks during operations if the simulator is not utilized 24/7. Additionally, simulators may connect to other systems that are also hosted on their own physical computers, requiring even more physical hardware to support the simulation-based training.

We aimed to reduce this hardware footprint by implementing a cloud computing approach that shares the utilization of hardware, storage, management, and associated costs while maintaining simulation-based training capability, systems availability, and allowing for upgrades. To accomplish this reduction, we prototyped the hosting game based training systems software and hardware resources on a cloud-based cluster. These technologies provide a collection of connected computer-based systems used to solve a wide range of computational tasks and calculations. Under this architecture, a computational job can be scheduled, assigned to a subset of the connected machines (nodes), processed,

and its results are returned to the user, all with the user-perception that a single system processed the job. The use of such a shareable, clustered system leverages the advantage of combining computational power and resources from multiple nodes, centralizes computer hardware, and is capable of automatically scheduling and sharing their resources with multiple users when needed. This methodology reduces the hardware footprints as users can share the clustered resources instead of duplicating the hardware purchases and presence, specifically addressing a core goal.

More specifically, the combination of virtualization, containerization, and clustering is being investigated to address the enhanced ITE's goal of having the synthetic environment reduce the hardware footprint. Virtualization permits a physical machine to be logically partitioned into multiple, virtual computers executing tasks in their own contained space, while clustering unifies the virtualization hardware into a single, centralized system. A software container is a virtualization method that allows one to package and distribute multiple isolated applications that share binaries and run under a common host operating system. The combination of virtual machines and containers being created and tasked through cluster scheduling utilizes compute resources more efficiently than individual servers being dedicated to execute one simulator. The efficiency exists because one physical machine can serve as multiple virtualized computers, where each virtual machine (VM) or container can host one or more simulators. Through virtualization, the cluster is able to schedule software to run on any supported operating system, regardless of the natively installed operating system of the cluster's nodes, including various distributions and versions of Linux, Unix, Mac OS, and Windows. Additionally, VMs can be allocated resources that more closely match their actual task needs, instead of dedicating an entire machine and having greater potential for resource under-utilization (Calheiros et al., 2011). Container technology can also serve as an alternative and/or supplementary solution to the use of VMs as it has been reported to perform nearly identical to a native performance (Felter et al., 2015). Containers are faster and smaller to create and migrate than VMs and provide a method of rapid simulation deployment and management. Through clustering, virtualization and containerization, this effort will identify methods to reduce the Army's simulation-based training hardware footprint.

Automated Simulation Event Scheduling

A second goal of the load balancing effort was to incorporate automated software scheduling into simulation event planning and deployment. Scheduling a simulation event is an essential function of training simulation environments as it may impact the execution of the simulation exercises. Scheduling is a labor-intensive task that involves analyzing capacity and performance needs, setting up storage, reconfiguring or restarting applications (Egts, 2014). Most military training exercises are still scheduled and configured manually using paper, whiteboards, calendars, and Excel spreadsheets to determine the availability of resources (Vidali, 2010). However, manual scheduling is prone to errors and is incapable of efficiently evaluating trade-off decisions when assigning available resources. When schedules are maintained on whiteboards and spreadsheets, it is not possible to collect accurate utilization metrics to support optimal resource load balancing during the planning process. Therefore, automation in simulation event scheduling is necessary in order to simplify the configuration, deployment, and management of the simulation applications and computational resources.

With the ITE being a cloud-based service that will host simulations at remote server locations, a simulation processing scheduler should be used to automate the simulation event scheduling process and assign a simulation to appropriate computational resources. The scheduler should also automate the deployment of simulation software, assign user priorities, and have the ability to preempt running simulation events. Fortunately, automated schedulers exist, but are not in use for LVC and gaming applications. For our investigation, we used Slurm, an open-source job scheduler that is used to automate the scheduling process of events and assigning computational jobs to clustered resources. The scheduler is currently installed on some of the fastest high performance computing system around the world for scheduling and provisioning research, academic, and commercial computing (Yoo, 2003) (Lucero, 2011).

Job schedulers, like Slurm, currently automate manually-labor intensive scheduling and hardware provisioning processes, facilitate planning, enforce user and job priorities, and provide visibility into resource availability. The software can be used to develop an intelligent system that will be in charge of scheduling jobs and provide automated deployment and management of the simulation applications. These resource managers can potentially assist in allocating computing resources needed for a specific simulation run by considering those needed by the simulation and those available, subject to user defined priorities.

Experiment Design

A series of experiments have been performed that progressively evaluate load balancing approaches in the modeling and simulation for training domain.

First, the experiments identified the benefits of using various approaches for automated simulation scheduling on the cloud. Second, methods for distributing the computational workload between cloud-based resources were prototyped and analyzed. Third, the feasibility of a reduction of the computer hardware footprint through the use of virtual machines and software containers has been demonstrated on a limited basis. Finally, techniques to dynamically manage the computational resources of those virtual machines and containers has been prototyped. The executed research plan first determined the feasibility of scheduling a game-based simulation on a cluster using Slurm. For this demonstration, we scheduled the execution of Microsoft Flight Simulator X (FSX) on a 2-computer cluster, where one computer served as the controller and the other as the node. With this system, we were able to schedule an FSX simulation event on the controller; then the node natively (directly on the node's Linux operating system) launched and displayed the game-based graphical, aviation simulator.

The second experiment scheduled multiple simultaneous simulations. For this study, two additional computers were added to the network and served as the client workstations (Figure 5). Here, two separate game-based simulation servers were scheduled with Slurm to run at the same time natively in a Linux environment. Once the simulations were running, one client workstation connected to one simulation and the other client workstation connected to the other simulation. The client workstations then displayed the graphics of their respective simulation and the user interacted with the simulated environment. This experiment demonstrated that a single physical server can host multiple simultaneous simulations and those simulations are accessible from separate client workstations. The next experiment iteration evaluated the feasibility of deploying the simulations through virtual machines and software containers, measured the computational overhead associated with each approach, and compared performance differences between scheduling the simulations natively, on a virtual machine, and in a software container. These experiments generated quantifiable comparisons on hosting multiple, simultaneous game-based simulations on a cloud-based system using the different load balancing approaches investigated in this effort. Furthermore, the experiments' generated data provided evidence that the computer hardware footprint for supporting simulations can be reduced with the use of virtual machines and software containers. Additionally, these and future experiments can demonstrate that automated software job scheduling can distribute computational workloads across available compute resources (load balancing), alleviate some of the challenges of manual simulation event planning, and can automatically enforce event, user, and group priorities.

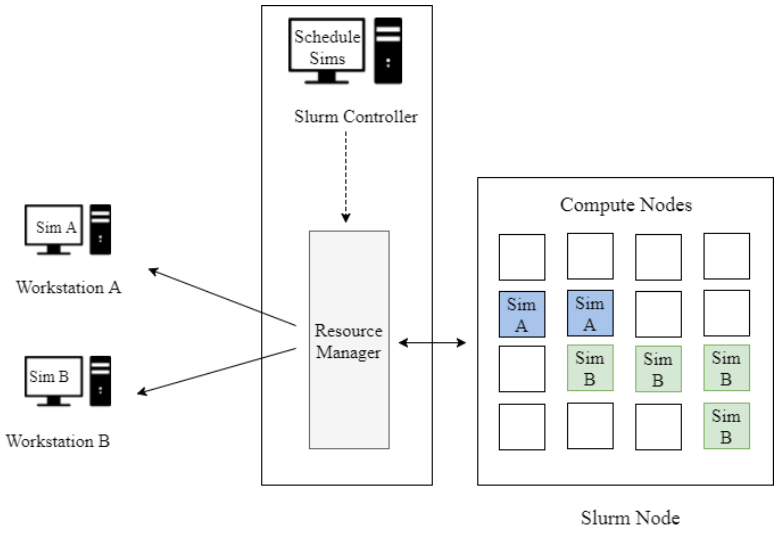

Figure 5. Experimental Setup for Scheduling Multiple Simulations Simultaneously

Future work

Future work will focus on automating a single scheduling system that can identify and start a series of private, connected (federated) regional clusters to remotely schedule, execute, and support simulation-based training on the cloud. Solutions will be investigated to address compute resource load balancing in two forms (a) on a single cluster and (b) across multiple geographically dispersed clusters. At the individual cluster level, simulation components will be deployed in containers that are hosted on one or more VMs. This approach will load balance jobs by reallocating resources to the VM when necessary. The second form of load balancing that will be examined is the distribution of compute jobs across multiple cluster federates (geographically dispersed clusters). A federated cluster will allow software jobs to be assigned to other clusters to evenly distribute workloads.

CONCLUSION

Our research produced valuable results by directly addressing the gaps related to the enhancements needed for a more robust ITE. We addressed the bandwidth and latency issues through the use of SDN-enabled architectures, allowing the Army to quickly provision and fully optimize their network connectivity. Further, we used thin clients as a complementary effort to our SDN work to reduce the hardware footprint while concurrently seeking innovative ways to train Soldiers using inexpensive thin client devices like Raspberry Pi through a cloud-based infrastructure. Finally, our load balancing efforts also focused on cloud technologies to reduce our hardware footprint and to further create a more efficient scheduling environment by using automated simulation event scheduling tools like Slurm. These three tasks will improve the ability to have a more efficient, optimizable, and flexible training simulation which will allow an enhanced versatility in terms of networking and content delivery.

ACKNOWLEDGEMENTS

We would like to acknowledge the work of the key IST staff members, to include Dr. Sean Mondesire, Dr. Anastasia Angelopoulou, Mr. Shehan Sirigampola, Mr. Chandler Lattin, Mr. Steven Zielinski, and Mr. Julian Montaquila. Their hard work and dedication to this project proved invaluable to our research.

REFERENCES

Calheiros, R. N., Ranjan, R., & Buyya, R. (2011, Sept.). Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments. In Parallel Processing (pp. 295-304).

DoD. (2016). *FY 2016 Budget Proposal*. Retrieved from https://www.defense.gov/News/Special-Reports/FY16-Budget.

Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2015, March). An updated performance comparison of virtual machines and linux containers. In *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium on* (pp. 171-172)

Egts, D. (2014). *How Linux Containers Can Solve A Problem For DOD Virtualization. Defense Systems*. Retrieved from https://defensesystems.com/articles/2014/07/02/egts-linux-application-containers.aspx

GameStream. Nvidia, Inc. Obtained January 18, 2017 from http://shield.nvidia.com/game-stream?utm_campaign=Oct_sale&utm_medium=Owned&utm_source=nvidia.com.

Harper, J. (2016). Army to Build Synthetic Training Environments. *National Defense Magazine*. Retrieved from http://www.nationaldefensemagazine.org/archive/2016/december/Pages/ArmytoBuildSyntheticTrainingEnvironments.aspx

Lucero, A. (2011). Simulation of batch scheduling using real production-ready software tools. *5th IBERGRID.*

The MITRE Corporation. "Point of Need Study for PM ITE: Evaluation of Experimental Cloud Hosting of Virtual Battlespaces 3 and Engagement Skills Trainer." Document Number MTR160007.

The MITRE Corporation. "Point of Need Study II." Draft.

Sheftick, G. (2016). Future VR Training Will Blend Global Partners, Air, Armor, Ground Forces. *TRADOC News Center*. Retrieved from http://www.nationaldefensemagazine.org/archive/2016/december/Pages/ArmytoBuildSyntheticTrainingEnvironments.aspx

Tolia, Niraj, David G. Andersen and M. Satyanarayanan. "Quantifying Interactive User Experience on Thin Clients." IEEE Computer, Vol. 39, No 3 (March 2006).

Vidali, A. (2010, 11). *Surviving the Perfect Storm: Ensuring Readiness with Integrated Training Technologies.* Retrieved from www.envisagenow.com

VBS3 Release Notes (Version 3.9.2). Obtained on January 18, 2017 from https://manuals.bisimulations.com/vbs3/3-9/manuals/#Release_Notes/Release_Notes.htm%3FTocPath%3DVBS3%2520Release%2520Notes%7C_____0.

Yoo, A. B. (2003). SLURM: Simple Linux utility for resource management. *Workshop on Job Scheduling Strategies for Parallel Processing* (pp. 44-60). Springer Berlin Heidelberg.