hTEC: A Layered MSaaS Architecture for Training and Experimentation Cloud

Erdal Cayirci
Research Center for S.T.E.A.M., FMV Işık University
Istanbul, Turkey
erdal.cayirci@isikun.edu.tr

Lutfu Ozcakir, Hakan Karapinar
Simulation Training and Test Sys Dep, HAVELSAN
Ankara, Turkey
ozcakir@havelsan.com.tr, hakank@havelsan.com.tr

ABSTRACT

A layered and distributed architecture for a training and experimentation cloud (hTEC) is introduced. The technical challenges of hTEC are discussed, and the solutions are explained. The hTEC layered architecture is aligned with the hierarchy and interrelations among the infrastructure, platform and software as a service models. The interfaces between the layers are also specified to complete the architecture. Prototypes for a subset of the hTEC services and interfaces are being implemented in a testbed called BSigma, where the hTEC architecture is evaluated. Preliminary results from our experiments are presented.

ABOUT THE AUTHORS

Erdal Cayirci graduated from Army Academy in 1986 and from Royal Military Academy, Sandhurst in 1989. He received his MS degree from Middle East Technical University, and a PhD from Bogazici University both in computer engineering in 1995 and 2000, respectively. He retired from the Army when he was a colonel in 2005. He was a faculty member and a researcher at Istanbul Technical University, Yeditepe University, Naval Sciences Institute and Georgia Institute of Technology between 2000 and 2005. He was Head, CAX Support Branch in NATO's Joint Warfare Center in Stavanger, Norway, and also a professor in the Electrical and Computer Engineering Department of University of Stavanger between 2005 and 2016. He is currently the Director of Research Center for S.T.E.A.M. in FMV Isik University. He received the "**2002 IEEE Communications Society Best Tutorial Paper" Award** for his paper titled "A Survey on Sensor Networks" published in the IEEE Communications Magazine in August 2002, the **"Fikri Gayret" Award** from Turkish Chief of General Staff in 2003, the "**Innovation of the Year" Award** from Turkish Navy in 2005 and the **"Excellence" Award** in ITEC 2006. He co-authored two books titled as "Security in Wireless Ad Hoc and Sensor Networks," and "Computer Assisted Exercises and Training: A Reference Guide" both published by John Wiley & Sons in 2009.

Lutfu OZCAKIR graduated from Hacettepe University Electronics Engineering Department in 1996. He received his MS degree from Bilkent University in Electronics Engineering Medical Signal Processing in 1998. He studies for a PhD on simulation of anatomical structures at Hacettepe University, Ankara. In 1998, he joined HAVELSAN, and worked as a team leader, the group manager and a project manager between 1998-2005, a project manager and the program director at HAVELSAN Simulation Systems Division between 2005-2011. He has been the Executive Vice President of Simulation, Training and Test Systems Division and a board member of HAVELSAN Technology Radar (HTR) since 2011. His research interests include Medical Signal Processing, Tactical Environment Simulation, Real Time Modelling, Distributed Joint Simulation Systems and C2 Simulation.

Hakan KARAPINAR graduated from Hacettepe University Electrical & Electronics Engineering Department in 1996. He received his MS degree from Bilkent University in 1998 about antenna simulation. He started working at HAVELSAN Company; Simulation, Training and Test Systems department in 1998 and he is still working as Program Director in that division. He studies for a PhD at Hacettepe University Electronics Engineering Department and an MBA at Cankaya University, Turkey. His research interests include Real Time Simulation, Modelling, Distributed Interactive Simulation, Electronic Warfare, Radar, Antenna, Tactical Environment Simulation and Sensor Simulation.

hTEC: A Layered MSaaS Architecture for Training and Experimentation Cloud

Erdal Cayirci
Research Center for S.T.E.A.M., FMV Işık University
Istanbul, Turkey
erdal.cayirci@isikun.edu.tr

Lutfu Ozcakir, Hakan Karapinar
Simulation Training and Test Sys Dep, HAVELSAN
Ankara, Turkey
ozcakir@havelsan.com.tr, hakank@havelsan.com.tr

INTRODUCTION

A highly scalable, layered and distributed architecture that supports interoperability, service discovery and composability is paramount in a service oriented cloud approach to modelling and simulation (M&S) for training and experimentation. Standardized services and simple interfaces to access them are required. The layered architecture, the services provided by each layer and their interfaces should be agreeable and amenable by all the stakeholders, including academia, industry, and the user community. An overly complex and centralized approach needs to be avoided to promote standardization and sustainability. It is preferable that the architecture supports the adoption of the previously developed services for the M&S as a service (MSaaS) ecosystem. A NATO Modelling and Simulation Group Technical Activity, namely MSG-136, is facilitating stakeholders from over 20 nations to achieve these goals.

MSaaS (Cayirci 2013) (Siegfried, Berg, Cramp and Huiskamp 2014) offers many advantages. A layered MSaaS architecture, such as Havelsan Training and Experimentation Cloud (hTEC) depicted in Figure 1 can promote reusability, interoperability and flexibility. hTEC follows an approach similar to Open System Interconnection (OSI). Any service can receive services from the lower layers through simple and standardized interfaces, and provide services to the higher layers. Therefore, in hTEC, it is possible to compose a simulation service mashup made up of models with various resolution and fidelity levels.

In Figure 1, the hTEC layers and their mapping to cloud service models including MSaaS (Cayirci 2013) is illustrated. The bottom layer in hTEC is a platform as a service layer (PaaS). In our test bed called BSigma, Armada, which is a HAVELSAN product, is used as PaaS. All the details related to the infrastructure and platforms are autonomously taken care by the PaaS according to the quality of service requirements specified by the higher layers.

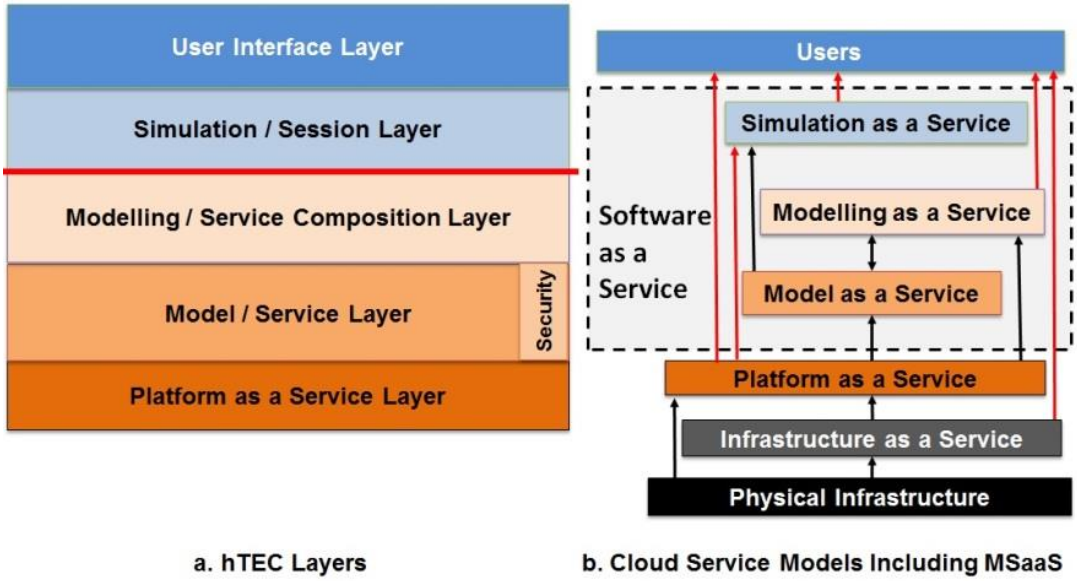

Figure 1. Mapping of hTEC Layers to Cloud Service Models including MSaaS

The service layer runs on top of the PaaS layer. The models in this layer manage and process the data related to the synthetic environment by using the services from Armada. The service layer provides models as services (MaaS) (Cayirci 2013), including database management functions. The users can manipulate the synthetic environments by using the services provided by the service layer. Please note that the security service is a sublayer within the service layer.

The service composition layer can compose a service mashup from the models provided by the service layer. It can be mapped to modelling as a service in cloud service models with a difference. Modelling as a service can be used to create new atomic or composed models (Cayirci 2013). In hTEC, the service composition layer (Cayirci 2013b) is not used for creating new atomic models but models composed of the services provided by the service layer. Please note that, when service composition is complete, a composed model, or in other words a simulation application (i.e., software) is compiled. Therefore, the layers below the red line in Figure 1 are before the compilation of a simulation application, and the layers above the red line provide run time services.

The session layer in hTEC runs the models composed by the service composition. Therefore, it is equivalent to the simulation as a service model (Cayirci 2013). It enables users to run multiple instances of the composed services or even federating them by using various interoperability technologies such as high level architecture (HLA) (IEEE 2010). Each instance runs with its own image of the synthetic environment, therefore the master copy of the synthetic environment is preserved for the usage of the others as long as needed. The instance management service also provides the users with the capability to run each of these instances as different types of simulations such as time stepped, continuous, static or dynamic.

The instance service can also decide on the parts of the services that need to be run in the front end due to stringent end to end delay constraints. The part of a MaaS with stringent delay constraints is called as the cerebellum function of the service (Cayirci, Karapinar and Ozcakir 2015). Cerebellum functions are migrated to the machines close enough to the front end (i.e., the machines that satisfy the delay constraints) by the PaaS layer.

In Section 2, hTEC architecture including the cerebellum function is explained in detail, where we also elaborate on the challenges and solutions. In Section 3, BSigma test bed is introduced and the preliminary results from the experiments are reviewed. In the same section, we also present our experience with virtualization, which is an important technology for cloud computing. Finally, the paper is concluded in Section 4.

hTEC OVER BSIGMA

In Figure 2, the examples for the services in each hTEC layer are illustrated. hTEC is designed as a distributed architecture. Therefore, there may be thousands of services available around the world when it is implemented as a public cloud. The hTEC architecture can also be used in a private cloud model where hundreds of services are available. Hence, service discovery and service composition is the first challenge. Please note that the interfaces between the control layer and applications are called as the northbound interfaces in software defined networking (SDN), and SDN Composition and Session Applications in Figure 2 are the hTEC services for SDN. We will further elaborate on that in this Section.

Service composition is a hard but solvable problem when a feasible solution that meets the criteria is sought instead of the optimum (Cayirci 2013b). In hTEC, the service composition does not have a time constraint because the services are wired into a single application during compilation, and then run in a machine in the cloud that satisfies the quality of service (QoS) requirements. As long as a standard approach is followed to define the services and to interface with them, service discovery is also a trivial task. There are already many standardized and scalable directory (X.500 2016) and service discovery (Helal 2002) mechanisms that can be used for this purpose.

The interfaces of the hTEC services are shown in Table 1, which has two parts: the meta data and the interface for the service. The meta data are the detailed and machine readable description of the service. It includes key information, such as, the service type, the fidelity, the resolution and the service model. The notation and the values for this information have to be standardized for interoperability. For BSigma purposes, we use a proprietary standard which is flexible. Please note that the first two fields in our structure are about the standard followed by the interface

and its version. Therefore, hTEC allows multiple standards in the ecosystem. However, for composing services that follow different standards, there will be a need for standard conversion before service composition.

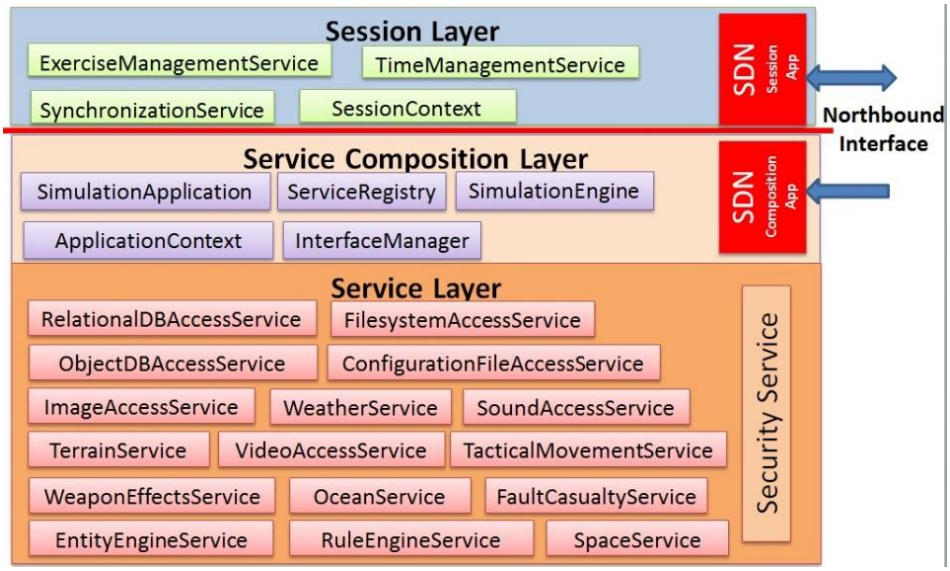

Figure 2. Examples for the Services in hTEC

The interface has three kinds of parameters similar to the structure of the subroutine calls in many programming languages: Please note that the name of the service is already among the meta data. That name is used for calling the service from inside the composed service. Apart from the name, the other fields in the interface are the return value and range, input parameter list including their types and ranges, and finally the output parameter list including their types and ranges.

Table 1. The Meta Data and The Interface for a Service

Type	Name	Remarks
Meta Data about the Service	The standard	The standard followed for the description and the interface of the service
	The version of the standard	The version of the standard followed
	The name of the service	The name of the service
	The service type	The type of the service (from the list in the standard)
	The resolution	The level of resolution (from the list in the standard)
	QoS Parameters	Values for the quality of service parameters (from the list in the standard)
	The fidelity	The level of fidelity (from the list in the standard)
	The description of the service	The details and important remarks about the service
	The version of the service	The version of this particular service
	The date	The date that this version is released
	The developer	The details of the developer
	The service model	Modelling as a service, model as a service, payment model and price, etc.
	The URL	The link for the service
	The cerebellum function	Null if none, the offset if the service has a cerebellum function
	The delay constraint	The distribution and statistics for the delay constraint
Inter face	Return Type	Type and range of the return value by the service
	Input Parameters	The input parameter list including the type and range of each of them
	Output Parameters	The output parameter list including the type and range of each of them.

Another challenge for hTEC is due to the propagation delay between the back end (i.e., the data center where the composed service runs) and front end (i.e., the machine used for interacting with the system). This becomes critical,

especially when interactive audio-visual systems are used. We developed a scheme called the cerebellum function to solve this issue.

The Cerebellum function includes the part of an MSaaS which is time sensitive in responding the user commands (i.e., inputs). Please note that the delay in responding to user commands by the simulation has to be the same as the delay in response to user commands by the real system. For example, if the delay in the real system d_r is between 90 and 100 msec, the delay in the virtual system needs to be within the same 90-100 msec window. As visualized in Figure 3, our scheme is based on the idea that the maximum delay between the user interface and cerebellum function d_{max} must be shorter than the lower bound of the real life system delay $r_{\alpha min}$ according to a given confidence level α. Hence, the delay can be managed such that negative training is avoided and immersion is maintained. The maximum delay d_{max} includes not only the propagation delay p_{max} introduced by the physical distance between two ends of a communications link but also computational delays c_{max} due to processes, such as encryption, decryption, routing, service federating, etc. We treat $d_{\alpha max}$ as a random variable, and make our computations based on the upper bound according to the given confidence level α.

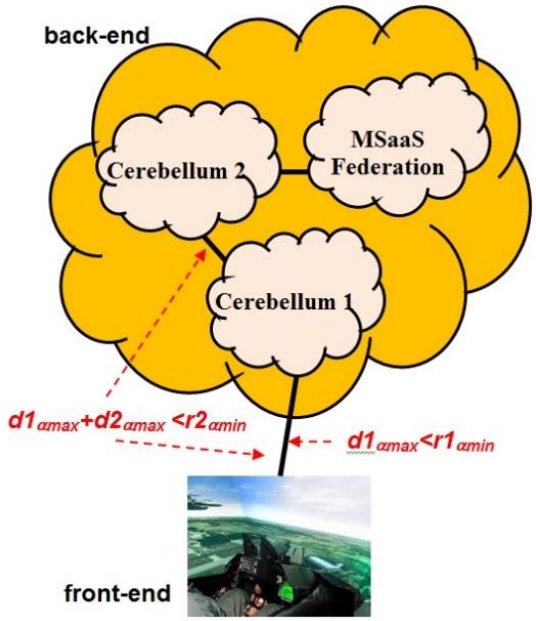

Figure 3. hTEC Cerebellum Function (d_1 is the simulation delay between the user and Cerebellum Function 1 and r_1 is the associated real system delay. d_2 is the simulation delay between the user and Cerebellum Function 2 and r_2 is the associated real system delay.)

When the services are designed, the designer should design the time sensitive part of the service as decomposable (i.e., can be separated from the rest of the service). Hence, the entire service and data does not need to be migrated closer to the front end but only the time sensitive part of the service. For example, the part of an interactive visualization service (IVS) that fetches the terrain data and weather conditions and creates three dimensional virtual environments can be designed separately from the part that makes the projections based on the user commands. The later part, which is time sensitive, becomes the cerebellum function for IVS. Please note again that this is only a simplified example to clarify the meaning of the cerebellum function.

In some cases, not only the cerebellum function of a service, but all of the service must be treated as a cerebellum function depending on the configuration of a composed service. If an input of Service s_a uses another Service s_b, which has a part that needs to be treated within the cerebellum function, s_a as a complete service has to be within the cerebellum function. Moreover, a cerebellum function may also have a nested structure, which means that the inputs of a cerebellum function may be coming from another cerebellum function. Therefore, the location of a cerebellum function is selected such that the conditions in Equations 1 and 2 are met.

$$dn_{\alpha\max} = \sum_{k=1}^{n} u(pk_{\alpha\max}) + u(ck_{\alpha\max}). \quad (1)$$

$$dn_{\alpha max} < u(rn_{\alpha max}). \quad (2)$$

where *n-1* is the number of cerebellum functions preceding the cerebellum function *n* in the nested structure. Please see (Cayirci Karapinar Ozcakir 2015) for the detailed description of the cerebellum function.

The cerebellum function can also provide better security for military MSaaS. Although the environmental data and specifications of military equipment, such as maximum speed and altitude that a military aircraft can reach are unclassified, the turn rates and similar data about the aircraft may be classified. Since the effects like turn rates are time sensitive and therefore will be typically treated by a cerebellum function in IVS, the cerebellum function approach may become useful also for dealing with the security related challenges of MSaaS because it stays in the front end.

In hTEC, two services (i.e., one in the service composition and one in the application layer) are introduced for software defined networking (SDN) (Hu, F., Q. Hao and K. Bao. 2014), namely the SDN composition and SDN session services. Both of these services are applications to provide northbound interfaces for the control layer in SDN as shown in Figure 4. The SDN composition application provides the service to retrieve the data about the network, such as the average delays between the nodes (i.e., hosts, switches and routers). These data are used for designing an SDN and determining the cerebellum functions and their locations during service composition. The SDN session application interacts with the SDN control layer to create and manage the designed SDN during the execution of the simulation

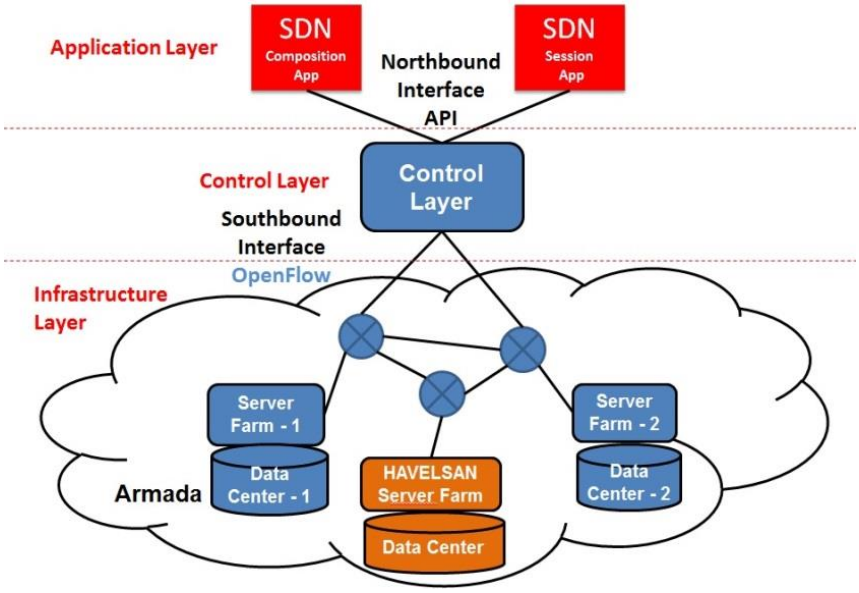

Figure 4. Software Defined Networking (SDN) for hTEC

EXPERIMENTAL RESULTS

In this section, the results from two different sets of experiments are presented. One of them is from NATO Computer Assisted Exercises (CAX). Since 2007, NATO CAXs in the Joint Warfare Center have been run in a completely virtualized environment. First, the findings from these exercises are summarized. Then, the results from the measurements for the hTEC architecture in BSigma are given.

NATO CAX support tools are run in a completely virtualized architecture during major exercises. These exercises were the first step for the realization of a service oriented simulation as a service (SOSaS) concept, and proved that all SOSaS services can be virtualized. Moreover, the virtualization of these services is more cost effective, easier to

prepare and administrate, and perform better than the conventional approach, i.e., not virtualized architecture. We summarize memory and CPU utilization data from NATO exercises below.

A typical virtualized architecture used during a NATO CAX is depicted in Figure 5. Six physical servers each with 32 GB of RAM and 1 TB of HD are used in this architecture. One of these servers is for backup. Three of the servers are for the simulation server processes. The other two servers are for virtual desktops. VMWare ESXi is used for the server virtualization and VMWare View is used for the desktop virtualization. In this architecture, 27 thin clients are used to provide the end users with CAX services. Each virtual machine for end users is dedicated 3 GB of RAM in our server pool. Before virtualization, 11 powerful servers with 16 GB of RAM and 512 GB of HD on average, and 27 powerful PCs used to be allocated for the same set of services.

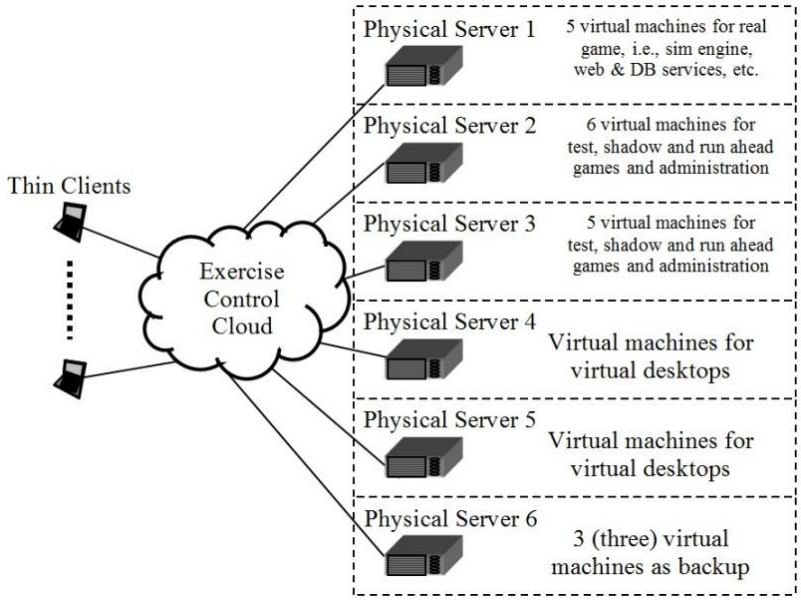

Figure 5. Virtualized Simulation Services during a Computer Assisted Exercise (CAX)

After testing this architecture in the first virtualized CAX in 2009, the number of servers is increased from 6 to 8, and one of the additional servers became the virtualized data center for the architecture. Please note that a five server architecture is already providing the required level of performance as shown in Figures 6-9. Additional servers are to further improve the performance and provide redundancy for fault tolerance.

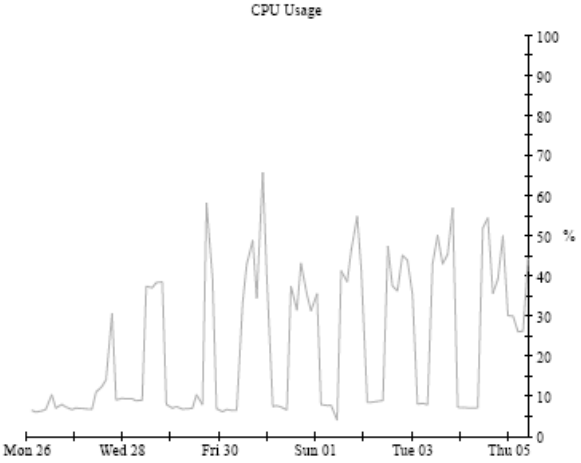

Figure 6. CPU Utilization of One of the Servers Used for VMWare ESXi

Figures 6-9 give the hourly averages of memory and CPU utilization for servers day by day throughout the exercise. Some parts of the graphs indicate 0% utilization. Those parts of plots are for night, when the exercise stops, and therefore servers are not utilized.

When the exercise is running, the CPU utilization of servers for CAX services is typically around 40%. The utilization is never close to 100%. The CPU utilization is flat, i.e., not bursty and in 35-55% band. On the other hand, the memory utilization is always above 85% but never over 95%. We can conclude that three powerful servers were sufficient to run the CAX servers comfortably. Please note that Joint Theater Level Simulation (JTLS) together with other CAX services, such as the joint exercise management module and C2 stimulation tools are run in this environment. The scenario is medium to high, which includes 10 brigades, 300 sea platforms and 1000 air sorties a day for one side in the average.

The utilization of the servers for the virtualized clients is different from the utilization of the servers for the virtualized servers as shown in Figures 8 and 9. The load created by virtualized clients is burstier. The utilization is sometimes close to 100% both for CPU and memory. Still it very seldom becomes a bottleneck and for only short time periods. Moreover, the users could hardly notice that.

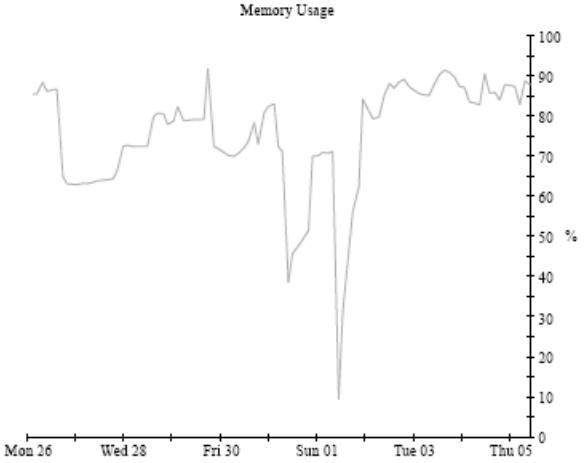

Figure 7. Memory Utilization of One of the Servers Used for VMWare ESXi

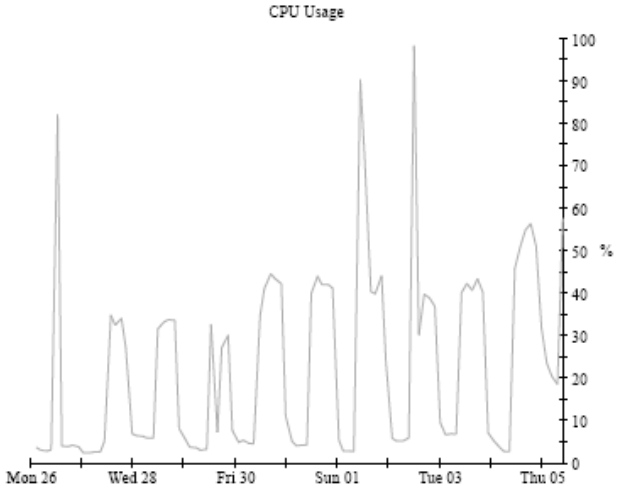

Figure 8. CPU Utilization of One of the Servers Used for VMWare View

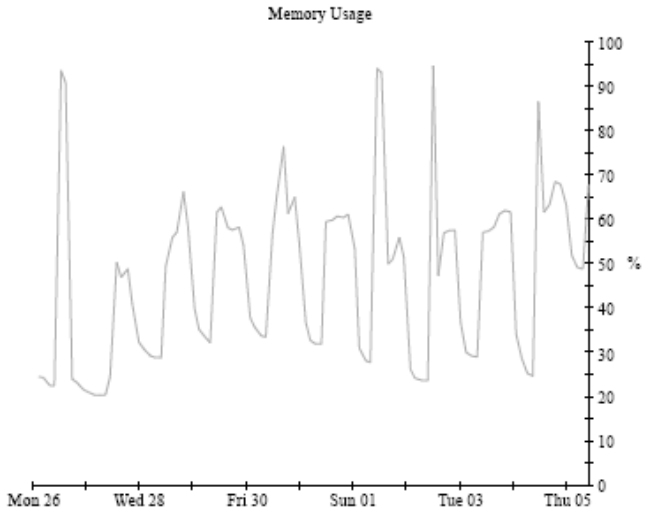

Figure 9. Memory Utilization of One of the Servers Used for VMWare View

In hTEC, we aim to reduce the hardware deployed to the remote sites (i.e., front end) significantly by keeping most of the infrastructure in the data centers (i.e., the back end) as shown in Figures 3 and 4. The front ends are used mainly to run cerebellum functions. Typically around 15 high end servers are deployed with the conventional full virtual simulators (i.e., aircraft, armored vehicle and submarine simulators). In BSigma, this capacity is reduced 80% in the average (down to two or three servers). When hTEC is complete, the cerebellum functions will be autonomously migrated to the front ends as designed by the SDN composition application.

In Figure 10, the actual response times to the controls (i.e., the response times of a helicopter to the controls) of a real helicopter and its virtual simulator are depicted. This shows that when the servers for a virtual simulator are at the site with the simulator, the realistic response times are achievable. Figure 10 also indicates that the simulator has to be able to start responding the controls within around10 millisecond in the average.

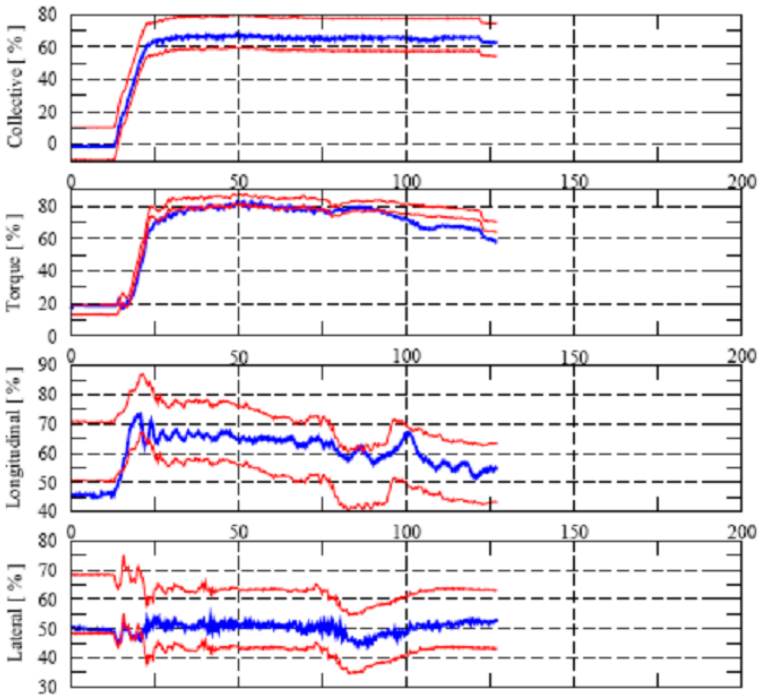

Figure 10. Response Times in mSec of Various Helicopter Controls (Blue is Actual Helicopter, Red is Minimum or Maximum Time for the Simulator.)

Figure 11 and Table 2 show the remote sites in BSigma and the propagation delays in the average between the hosts in these remote sites. These delays are very low, and we do not think that they can get lower in medium term. Please note that the processing delays are not included into the numbers in Table 2. The delays given in Table 2 are at least twice longer than the time constraints implied by Figure 10. This proves the necessity for the cerebellum function. We continue implementing hTEC and BSigma to test if our cerebellum function can provide realistic response times.

Figure 11. The Planned Sites for BSigma

Table 2. The Average Propagation Delay Among the Remote Sites in BSigma.

CITIES		A			B			C			D			E			F			AVERAGE TIME (ms)
	CLOCK	9:00	13:00	17:00	9:00	13:00	17:00	9:00	13:00	17:00	9:00	13:00	17:00	9:00	13:00	17:00	9:00	13:00	17:00	
A	Time (ms)				30,465	30,824	33,544	8,213	13,591	8,483	15,975	15,897	15,459	9,334	16,372	11,240	13,979	12,452	11,877	**18,396**
B	Time (ms)	29,425	30,570	35,407				36,954	39,158	40,474	42,666	44,031	43,693	24,304	26,092	25,329	35,754	32,495	35,224	**36,686**
C	Time (ms)	8,947	9,520	8,167	41,776	39,914	34,422				26,007	25,925	28,343	20,432	16,705	16,077	19,897	21,292	19,913	**24,413**
D	Time (ms)	20,157	18,158	17,182	44,711	44,596	44,509	34,367	25,826	26,503				24,187	25,632	24,439	24,997	25,651	24,310	**30,451**
E	Time (ms)	9,212	9,468	9,327	24,912	25,353	24,452	17,017	15,040	14,816	22,537	23,306	25,138				16,703	20,163	14,181	**20,407**
F	Time (ms)	17,216	19,701	15,044	35,783	34,273	35,559	19,928	25,632	20,304	25,375	25,172	24,187	15,589	14,618	14,615				**24,557**
AVERAGE TIME (ms)		**18,752**			**37,289**			**25,478**			**28,084**			**20,569**			**24,739**			

CONCLUSION

The MSaaS approach not only reduces the cost but also introduces many other benefits such as: flexibility in capacity and architecture, ease in management and licensing, need for fewer number of engineers and technicians, more efficient and reliable system and security engineering.

hTEC is our service oriented implementation of MSaaS, which is a layered architecture. Havelsan ARMADA provides PaaS for hTEC. The hTEC service layer is over ARMADA and provides various M&S services in the form of libraries. Security related services are treated as a sublayer within the hTEC service layer. The hTEC service composition layer selects a subset of the services from the hTEC service layer according to the requirements of the simulation, and wires them into a simulation application. The hTEC service composition layer has the SDN composition application which communicates with the control layer of the SDN to design a network and configure the cerebellum functions accordingly. The hTEC session layer is the topmost layer, which controls the SDN by using the SDN session application and executes the required number of instances of the composed hTEC application. The hTEC session layer is responsible also for federating the instances by using technologies such as HLA when required. The hTEC architecture is implemented as a testbed called BSigma. The preliminary results from the experiments in BSigma are encouraging.

ACKNOWLEDGEMENTS

We thank Nick Giannias for comments that greatly improved our manuscript.

REFERENCES

Cayirci, E. (2013). Modelling and Simulation as a Service: A Survey. *In Proceedings of the 2013 Winter Simulation Conference, edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, forthcoming. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.*

Cayirci E. (2013b). Configuration Schemes for Modelling and Simulation as a Service Federations. *Simulation Transactions of the Society for Modelling and Simulation International, Vol. 89, Issue 11, pp. 1388 – 1399.*

Cayirci, E., Karapinar, H., & Ozcakir, L. (2015). Cerebellum Function for MSaaS. *EMSS 2015*.

Helal, S. (2002). Standards for Service Discovery and Delivery. *IEEE Pervasive Computing, Vol. 1, Issue 3, pp. 95-100.*

Hu, F., Hao, Q., & Bao, K. (2014). A Survey on Software Defined Network and OpenFlow: From Concept to Implementation. *IEEE Communications Surveys and Tutorials, Vol, 16, Iss 4, pp. 2181-2206*.

IEEE, (2010). 1516-2010 - IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)--Framework and Rules.

Siegfried, R., Berg, T., Cramp, A., & Huiskamp, W. (2014). M&S as a Service: Expectations and challenges. *SISO 2014 Fall Simulation Interoperability Workshop, Paper 14F-SIW-040, Orlando, USA*.

X.500, (2016). Retrieved April 2016, from http://www.x500standard.com/