THIN CLIENT DISTRIBUTED SIMULATION OF DISCRETE EVENT MODELS

Colin Timmons, Gabriel Wainer

Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada

{colintimmons, gwainer}@sce.carleton.ca

ABSTRACT
In recent years, new techniques have been created to allow practitioners to model real world events using either military training environments or discrete event simulation tools. Military distributed simulations are interoperable through interfaces such as High Level Architecture (HLA), or Distributed Interactive Simulation (DIS) Protocol while discrete event Modeling and Simulation (M&S) are defined through their tools and formalism. The RESTful Interoperable Simulation Environment (RISE) is a plug and play paradigm of software tools that has a resource application interface based on the Uniform Resource Identifiers (URIs). We show how these diverse branches of simulation can be joined together to produce a mash-up of services for distributed simulations allowing precise discrete event modeling in a military context.

Keywords: DEVS, Cell-DEVS, Discrete Event Models, Distributed Simulation, HTML5, military, model, DIS, Restful-CD++, simulation, WebLVC

1. INTRODUCTION

In the logical flow from doctrine through training to operations, the intent of the military is to establish a force generating system that is responsive to operational demands. Training simulation technology is making significant advances in flexibility, fidelity, interoperability, networking, portability, reliability, and fiscal effectiveness. The aim of the training is to expose a soldier to various battlefield conditions and environments that will allow the soldier to develop, learn, and confirm skills. Training simulation technologies can replicate battlefield conditions and equipment platforms to a high degree of fidelity with low cost and are collectively interfaced together as a Live, Virtual or Constructive (LVC) distributed simulation.

Military simulations can exist as stand alone training platform simulations or stimulators to highly inter-connected real-time functional-level or platform-level war-gaming. In the context of war-gaming, these distributed simulations are connected through interoperable standards such as the following (NATO 2013):

1. High Level Architecture (HLA): HLA is defined as “a family of related standards that together describe the functional elements, interfaces, and design rules for a unified approach and common architecture to constructing interoperable simulation systems”;
2. Distributed Interactive Simulation (DIS): DIS is defined as “the protocol used for information exchange between Synthetic Environment (SE) components”; and
3. Protocol or Testing and Training Enabling Architecture (TENA): TENA is defined as “a distributed simulation communication protocol, described as middleware, and designed to enable interoperability among range systems, facilities, simulations, C4ISR systems in a quick, cost-efficient manner”.

With a theatre level simulation, a soldier experiences the friction, stress and uncertainty of virtual combat without the steep cost associated with training in a live combat situation. At the same time, the soldier’s performance can be monitored, recorded, analyzed and evaluated in order to measure the level of efficiency. Battlefield lessons usually learned painfully and only at the start of a deployment campaign can now be learned during peace time allowing effective troop readiness for deployment.

Within the operational theatre, planning processes use simulation to advance the pre and post planning cycles of combat situations. Modeling and Simulation (M&S) in theatre focuses on the experimentation of known circumstances and conditions in order to justify and validate the tactics and resources required to succeed. Portable modular classrooms allow battle space training in theatre or on the fly. However, these portable modular classrooms can only be remotely controlled through DIS or HLA gateways.

Research has been conducted to provide traditional modeling paradigms with composed and hierarchical integrated multi-models that can dynamically change their structure or behaviour. Cell-DEVS can generate this complex behaviour from sets of relatively simple underlying rules (Ameghino, Troccoli and Wainer 2001). Using this technique, emergent behaviour in a

complex adaptive system is generated without the need to include centralized control mechanisms or equations. Bottom-up rule conditionality allows the interaction of higher level components. Establishing an experimental framework is critical to the verification and validation of a model as it allows the user to precisely evaluate a simulation run with varying input parameters or conversely, many simulation runs with many variable input parameters (Al-Zoubi and Wainer, 2009).

The RESTful Interoperable Simulation Environment (RISE) middleware is a REST based server that is a plug and play software paradigm of software tools and whose resources are accessed through its resource Application Interface (API) as Uniform Resource Identifiers (URIs) (Berners-Lee 1996). RISE, following the requirements of statelessness, allows a user to conduct M&S through the resources named by the URIs. In doing so, RISE encapsulates the technical functionality and provides a platform that is independent of formalism and tools.

HyperText Markup Language - version 5 (HTML5) is a family of web technologies currently being specified by the World Wide Web Consortium (W3C) and includes new HTML markup tags, Cascading Style Sheets - version 3 (CSS3), JavaScript and other supporting technologies. These other technologies include Geolocation, WebSockets Web Workers, and local and web storage. Advances of this technology have allowed the development of more useful and sophisticated webpages (Pieters 2013).

With the opportunities created by HTML5, advances have been made that allow web-based clients to access the traditional distributed simulations. Leveraging the new WebLVC protocol that harnesses the power of HTML5, a thin client can be developed that creates a means to leverage web-based simulations and provide a web-based mash-up of services for the traditional distributed simulations.

2. BACKGROUND

2.1. Distributed Simulation Interoperability

Many technologies have been developed in order to provide an interoperable methodology to interface and integrate real-time functional-level and platform-level war-gaming. Originally a prototype research system, the SIMulation Networking (SIMNET) application grandfathered many of today's dynamic implementations such as Z-Buffering and network protocols. The use of SIMNET during the Gulf War demonstrated the success of a real-time interactive networked cooperative virtual simulation (Wilson and Weatherly 1994). DIS and the Aggregate Level Simulation Protocol (ALSP) replaced SIMNET with the purpose of allowing dissimilar autonomous simulation nodes to interoperate in real-time, interactive, distributed simulations (Smith 1995). Advances in technology led to a follow-on architecture called HLA which builds upon the DIS effort by the DIS Steering Committee in 1994 and the ALSP.

Originally conceived as the interoperable methodology for distributed simulation, the Department of Defense (DoD) mandated the establishment and requirement for interoperability of all its simulations through the use of HLA. Desiring to leverage the advantages of HLA, the North Atlantic Treaty Organization (NATO) ratified their M&S Master Plan in 1998 and identified HLA as the interoperability standard. Further, in order to provide input into the evolution of HLA, NATO required DoD to evolve the HLA standard through the Simulation Interoperability Standards Organization (SISO). However, by relaxing the policy for the requirement for the adoption of HLA, transition to HLA became the decision of DoD components' priorities, requirements, and resources. DoD's neglect to mandate and enforce HLA as the only standard provided the avenue for other distributed simulation architectures and protocols such as TENA, and the Combat Training Instrumentation System (CTIS), Tactical Data Links (TADIL), Command, Control, Communications, Computers and Intelligence (C4I) and the survival of the DIS protocol (Hollenbach 2009).

The majority of these interfaces are developed by different vendors, requires different techniques for achieving their functionality, and requires technical Subject Matter Experts (SME) to install, configure, test, operate and maintain. Leveraging different monolithic applications together to create LVC simulations for the different DoD training domains has resulted in multiple types and instances of protocol translators to integrate the monolithic assets. Research has been undertaken to reduce the number of these protocol translators (Bizub, Bryan, and Harvey 2006).

However, common to all these interfaces and their application, is the requirement for a protocol or architecture framework that allows the monolithic applications to interface. The data passed is an entity which is defined as "any distinguishable person, place, thing, event or concept about which information is kept" (SISO 2007). Any simulation designed without these specified architectures or protocols are not interoperable in a real time functional-level or platform-level war gaming distributed simulation. Examples of such simulations could be a flight simulator, a combat information centre, or a web-based simulation.

2.2. Model Development on RISE

The key goals of Representational State Transfer (REST) is the scalability of component interactions, its generality of interfaces, its independent deployment of components and its intermediary components to reduce latency delays, compel security implementations and encapsulate legacy systems (Fielding 2000). RISE is designed under this architectural style and has its service URIs structured in a hierarchal tree as shown in Figure 1.

As stated previously, RISE is developed as a plug-and-play paradigm of software tools whose resource API is accessed through URIs. The state changes of the server are issued upon the user transiting through the URIs to access its different resources. The user navigates the URIs from a thin client application such as a web browser and accesses the resources through the HyperText Transfer Protocol - version 1.1 (HTTP/1.1) methods as defined under RFC 2616. HTTP's methods DELETE, GET, POST, and PUT are a feature of the negotiation of data representation which allows systems to be built independently of the data being transferred.

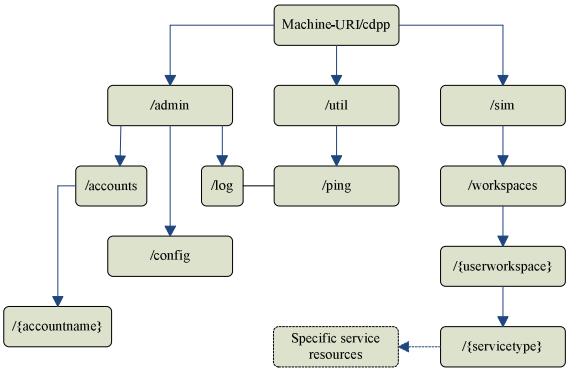

Figure 1: RISE Common Resource Structure

RISE is a platform and service that was implemented to provide server resource availability for client applications. Specifically, RISE was created as a middleware resource for a RESTful implementation of the Distributed CD++ (DCD++). CD++ is a toolkit for discrete event M&S and is based on the Discrete EVent System Specifications (DEVS). The design modeled into the RISE middleware was to provide transparent sharing of computing power, data models, and heterogeneous environments on a global scale. RISE provides a lightweight approach to web services. It hides internal software implementation as compared to the Simple Object Access Protocol (SOAP) based web services, which rely on Remote Procedural Calls (RPCs). The simulation data hosted on RISE is Cell-DEVS (Al-Zoubi and Wainer, 2009).

2.3. New Technologies and Standards

New technologies and standards have developed for the World Wide Web (WWW) which enables highly interactive, low-latency, real-time web-based applications. HTML5 is the newest version of the markup language used for structuring and presenting content for the World Wide Web. This revision is still under development though most modern browsers are fully compatible of supporting the version. HTML5 was designed to provide a single markup language between HyperText Markup Language (HTML) and eXtensible HyperText Markup Language (XHTML) while also implementing other new elements such as audio, video and canvas elements to name a few. These new elements are designed to handle graphical multimedia rich environments without having to resort to third party plug-ins (Pieters 2013).

The HTML5 File API allows the read-only capability of the user's system. As a working draft of the File API specification, the HTML5 File API was expressly designed to allow web applications the ability to access files which a user may upload to a remote server or manipulate in a rich web application. Designed to remove the reliance on third party embedded APIs, HTML5 standardizes the way to interact with a local file. As RISE presents its simulation results resource as a compressed file for its event transition, this file access becomes critical to developing a mash-up of services (Ranganathan and Sicking 2012).

Implemented in HTML5 are WebSockets. WebSockets allow the bidirectional transportation of the JavaScript Object Notation (JSON) objects through communication channels called WebSockets. They are somewhat a combination of UDP and TCP in that they pass messages like UDP but have the reliability of TCP. With a combination of the two protocols, a client is able to create an asynchronous full-duplex channel to the host server. This communication allows the client to send data instantly to the server and have the server communicate to the client concurrently while the connection is open.

WebSockets, besides defining a new protocol for the transference of data, also provides a method for creating secure connections (Lubbers and Greco 2009). Similar to normal asynchronous calls like TCP, where the protocol is optimized for accurate delivery rather than timely delivery, all bytes received will be identical to all bytes sent and in the correct order. WebSockets do not have the problem of TCP as TCP incurs relatively long delays while waiting for out-of-order messages or retransmissions of lost messages using its positive acknowledgement technique. This TCP technique requires the receiver and sender to send an acknowledgement message each time it receives data segments, preventing the streaming of data. Critical to the accurate delivery technique of TCP is that the sender is required to keep a timer on the transmitted packet so that if an acknowledgment message is missed, the timer will expire and the transmitted data is resent. Once the full data is transmitted, the receiver no longer talks to the sender.

Previously, thin clients such as browsers operated sequentially. In this regard, there is only one User Interface (UI) thread that processed and manipulated all the data within a web browser. Though the multi-threading of web workers are in its infancy, client applications are no longer single-threaded and must rely on server generated state information. Processing can now be off-loaded back to the client, freeing the server from computational requirements and state changes. With web workers, thin clients are now capable of processing like modern applications with multithreading

in that they can produce multiple threads that allow data manipulation and calculation in JavaScript (Hickson 2012).

To allow web-based simulations access the traditional distributed simulations, MÄk has leveraged their experience to aid in the development of single entity simulations (MÄk 2012). Prototyping the protocol and submitting an initial draft to SISO for study, MÄk has provided the framework for a consensus-based interoperability standard using the JSON objects matched with the built in encodings for DIS and Real-time Platform Reference Federation Object Model (RPR FOM) semantics. MÄk's WebLVC server permits the protocol to define a standard of passing simulation data between a web-based client application and a WebLVC server while remaining independent of the protocol used within the distributed simulation. Hence, a web-based client application using the WebLVC server could participate in distributed simulation exercise such as a DIS exercise, an HLA federation, a TENA execution or any other distributed simulation environment.

3. FRAMEWORK FOR SUPPORTING ROBUST DISTRIBUTED SIMULATIONS

This section introduces the design of the thin client which manipulates the RISE simulation execution results file into a format acceptable for dissemination to a distributed simulation. No implementation can guarantee total coverage for fault isolation and the focus of this study was to develop a robust infrastructure that would allow discrete event simulation such as those available on RISE to facilitate injection of valid model simulation execution results into a distributed simulation. It assumes that the thin client used has the minimum of capabilities such as WebSockets, file access and multi-threading.

3.1. Thin Client Distributed Simulation of Discrete Event Models Architecture

Access to distributed simulations has not been established by web-based architectures because of the limited capabilities of previous versions of thin clients and the latency of reliable data throughput.

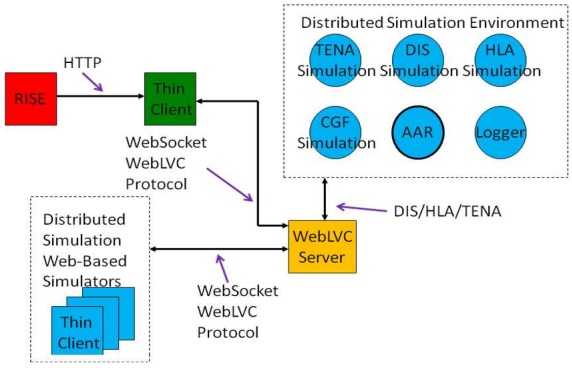

Figure 2: Thin Client Distributed Simulation Of Discrete Event Models Architecture

However, with the combination of new technologies and standards, monolithic applications present within a distributed simulation can now be interfaced by the WebLVC server. The server stands connected to the distributed simulation similar to a DIS/HLA gateway but receives and transmits JSON-based data to web-based simulations through its WebSockets. Combining this technology and methodology, we directly interface and integrate discrete event simulation data as presented on RISE into a distributed simulation regardless of the type of distributed simulation environment. This is illustrated in Figure 2.

3.2. WebSocket Benefits

Use of the WebSocket protocol permits bi-directional data exchange with the streaming capability of UDP and the reliability of TCP. It also allows persistent connections between the client and the server by providing the reception of responses without polling or need for requests. Additionally, the use of WebSockets is efficient in that its header is minimal at two bytes for small frames and up to ten bytes for large frames, which compared to HTTP header traffic of up to 2000 bytes is significant (Lubbers and Greco 2009). In comparison to TCP, which enables the streaming of bytes, WebSockets enables the streaming of messages and therefore reduces unnecessary network traffic, bandwidth consumption and latency. In this regard, WebSockets are competitive with the bidirectional communication channels of distributed simulations as shown in Figure 3. In this case of a HLA distributed simulation, data is passed between the federate and the Real-Time Infrastructure (RTI), which is responsible for receiving and sending the data to other component simulations through its callback methods.

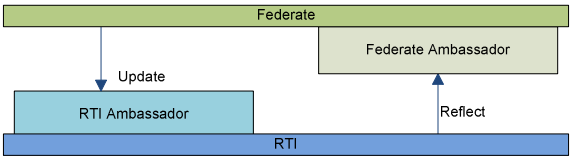

Figure 3: Bidirectional Distributed Simulation Communication Channels

3.3. Multi-Threaded Application

Most applications spawn separate threads for concurrent processing either by time-division multiplexing on single core processors or by having every core or processor executing a separate thread simultaneously on a multiprocessor or multi-core system. With web workers, thin clients such as a browser, no longer are restricted to sequential processing. Parent threads such as the UI thread of a web browser can spawn child threads for concurrent processing. Similarly, child threads can also spawn new child threads. However, as data is shared by the threads, web workers have been designed not to be able to access the Document Object

Model (DOM) because of thread-safety, the window object, the document object, or the parent object. It also does not have access to create and bind a WebSocket. It does, however, have access to the navigator object, a read-only capability of the location object, XMLHttpRequest, timers, and the application cache. Aiding in the modular development, threads can also import external scripts providing a maintainability aspect to their production and implementation (Hickson 2012).

Concurrent operation by threading increases the power of the thin client when combined with the advantages of HTML5. HTML5 was designed to provide a single markup language between HTML and eXtensible HyperText Markup Language (XHTML) while also implementing other new elements such as audio, video and canvas elements to name a few. These new elements are designed to handle graphical multimedia rich environments without having to resort to third party plug-ins. The HTML5 canvas element allows the client web browser to inherently manipulate, construct and layer image data such as 2D or 3D images (Ranganathan and Sicking 2012). The thin client uses the Environmental Systems Research Institute Aeronautical Reconnaissance Coverage Geographic Information System (ESRI ARCGIS) street map and the ESRI ARCGIS Imagery World 2D map for user verification of the entities.

The file selection mechanism of the HTML5 File API allows the read-only selection of a simulation results file selected by the user on RISE for manipulation and publishing into the distributed simulation. As an example of the benefits of multi-threading, the RISE simulation results file was decompressed and data was extracted from the specific textual log file. The concurrent action of decompressing and data extraction is readily apparent by the use of threads. Decompression and extraction processing requirements occurred as background processes during the image loading times of the document. Manipulation and calculations were processed concurrently with the user's selection of the simulation results compressed file.

3.4. Data Extraction and Manipulation

Distributed simulations require unique identifiers for any existing entities which are defined as any component in a system that requires explicit representation. Such a requirement necessitates examination of the possibilities of injecting data into the distributed simulation. One option was to inject the discrete event simulation as a single entity, or conversely, each populated cell in the Moore's neighbourhood of the Cell-DEVS simulation data becomes an entity. The latter method would allow the depopulation of the cells along with the population of cells. In order to accomplish this requirement, elimination of any duplicate lines in the text log file was mandatory.

The simulation data hosted by RISE is formatted in such a manner that data extraction is dependent on the format of each textual line. As each element in the log file is separated by a white space, the element is extracted and stored in containers indexed by the line number of the text file. Positional reference of the cell in the Moore's neighbourhood is based on the Cell-DEVS grid cell origin and can be selected as a latitude and longitude origin. Distance and bearing is linearly calculated for the specific cell from the grid cell origin. This is illustrated in Figure 4. Scaling of the distance and thus altering the bearing on the grid cell is selected by the user and calculations are concurrently performed during the file selection process which has the greatest delay because of the destruction of the modal open /save dialog. Because the Cell-DEVS discrete event models are oriented from a heads up perspective, the models are not oriented to the mapping structure of the ERSI ARCGIS maps.

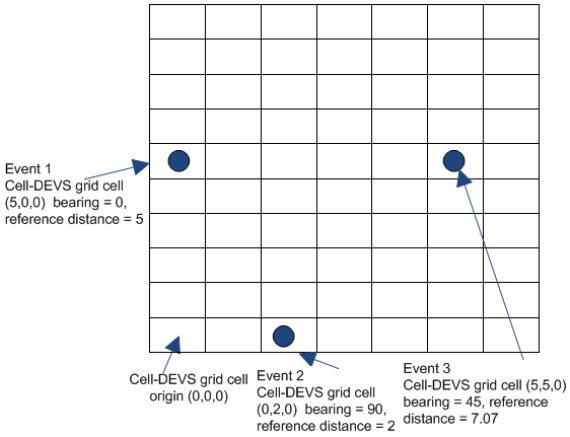

Figure 4: Cell-DEVS Moore's Neighbourhood with Positional Bearing and Distance

3.5. Time Management

Cell-DEVS operates as a sequence of distinct events where each event marks a change of state in the system. In this capacity, the simulation results may have zero up to the grid size of the Moore's neighbourhood in state changes occurring at every event. Problematic in the event space is the time period of execution and its correlation to real-time. Execution time of the simulation is usually minimized in its measurement units to benefit the Cell-DEVS modeler. For example, though a real-world system process may take days to occur, a Cell-DEVS modeler will model and simulate the system with a time measurement unit of milliseconds, or conversely, the simulation time occurs in days when the desired real-time is in hours. Distributed simulations are real-time simulations where the simulated time advances at the same rate as real time, thus a scalability factor is required to advance the sequence of events into the distributed simulation to control the speed of the population of entities.

Actual control of the population and removal of the entities into the distributed simulation is user selectable through command button functionality. This command button functionality also commands the thin client's variable states and creation point for the child manipulation processing threads.

3.6. WebLVC

There is currently no standard interoperability protocol to adequately link new web-based applications with each other and with traditional distributed simulations. MÄk's WebLVC protocol attempts to correct this lack of interoperability. The company's WebLVC server has become the bridge through which web based simulations interoperate with distributed simulations over the WebLVC protocol while remaining transparent to the intricacies of the distributed simulation interoperability standards. The type of JSON message interchanged with the WebLVC server can be the following: 1.Any, 2. Other, 3. Attribute update, 4. interaction, 5. connect, or 6. object deletion.

For clarification and differentiation between an attribute update and interaction message, an interaction message would inform the distributed simulation applications that a F18 is circling with a radius of 1000 around a specified world positional coordinate while the attribute update is the default type of message for instantiating and updating an entity. The draft specification of the WebLVC protocol, version 0.1, only documents the attribute update (MÄk 2012).

As the WebLVC protocol is still in development and being presented to SISO as a standard, work still needs to be accomplished with regard to the distributed simulation interoperability standards. Specifically, as the WebLVC protocol is a JSON-based wire protocol and leverages on creating non Federation Object Model (FOM) specific clients, work has yet to be completed on the WebLVC-to-HLA mappings and the HLA-to-WebLVC mappings; however, the DIS protocol capability is instantiated.

3.7. Entity Generation Implementation

Enclosed within the JSON entity is the specifics of the entity which uniquely describes the entity. In order to allow diverse entity types, the thin client provides an alterable numeric enumeration array that specifies the entity type. In Figure 5, an example of the JSON entity message is illustrated and shows the required attributes of an entity within a DIS environment. Its size is dependent on the attributes size but in this case was no more than 448 bytes. The attributes such as entity type, marking, entity identifier and object name are selectable and can be altered prior to selecting the compressed simulation results file from RISE. These attributes may also be entered as a query string upon the loading of the thin client.


```
{
"AccelerationVector": "[0,0,0]",
"AngularVelocity": "[0,0,0]",
"DamageState": 0,
"DeadReckoningAlgorithm": 4,
"EntityIdentifier": "[1,1,920]",
"EntityType": "[4,1,0,0,0,0,0,0]",
"ForceIdentifier": 0,
"Marking": "WebLVC-CDpp",
"MessageKind": "AttributeUpdate (1)",
"ObjectName": "CDpp 69354_9_2_0",
"ObjectType": 1,
"Orientation": "[1.818082012506282,
                -0.7780143250425958,
                -3.1398729077441]",
"Timestamp": 1358.53041601181,
"VelocityVector": "[0,0,0]",
"WorldLocation": "[1108268.601076215,
                  -4345117.870447309,
                   4520465.02507074]"
}
```


Figure 5: WebLVC Protocol JSON Entity

A deletion message which informs the WebLVC server to remove entities from the distributed simulation is characterized by the entity identifier, message type and the entity's unique object name as shown in Figure 6.


```
{
 "EntityIdentifier": "[1,1,28270]",
 "MessageKind": "ObjectDeletion (4)",
 "ObjectName": "CDpp1_42942_28_27_0"
}
```


Figure 6: WebLVC Entity Deletion Message

3.8. Experiments and Results

In order to verify the correctness and investigate the performance incurred in the proposed thin client manipulation and parsing of Cell-DEVS data, a series of investigations was conducted to detail the robustness and the performance of the thin client platform. The hardware platform used for the experiments was a custom built Intel platform. The computer housed a Z68 motherboard containing 16 Gigs of memory, a K7-2600 CPU, a Radeon HD 6850 video card and a gigabit Ethernet connection. The network was connected to a high-speed cable modem via CAT6 Ethernet cable. The WebLVC server was hosted on a server containing 2 Gigs of memory, a Pentium 4, and a gigabit Ethernet connection.

Paramount to any application is its responsiveness. The de facto standard from a user's perspective for delay is 300 milliseconds. In order to minimize the effect of browser delays, the thin client harnessed the Dojo API library to provide Asynchronous JavaScript and XML (AJAX) capability so that processes to do not depend on other processes' outcomes. Similarly, Cell-DEVS data manipulation and processing were offloaded into child threads in order to provide efficient concurrent operations. In particular, the manipulation of data was designed to provide the user with the option of selecting an epoch or non-epoch process.

Non-epoch processing, when selected, manipulated and parsed the data as a background thread concurrent with

the Dojo event handler. However, data generation of the entities remained available to the UI thread in the case where the time period of the simulation execution is minimal, thus requiring a high demand of messages to be passed between the threads. Epoch processing, conversely, utilized the concurrent processing of the threads to create the entity data as timings of the messages passed between the threads was advantageous.

Function	Mean (ms)	Std. Deviation	Confidence Interval (95%)
Multi-layer Creation Function	396.54	17.19	12.28
File Selection Function	402.60	21.29	15.22
RestfulResource initComms Function	19.00	1.25	0.89
Start Button Function	16.60	3.06	2.19
Zoom To Scale Function	3.40	1.17	0.84
Change State Function (start)	24.00	4.47	3.20
Entity Initialization Function	4.40	1.65	1.18
CDppEntity Initialization Function	1.00	0.47	0.34
Create Entity Graphic Function	2.20	0.92	0.66
Set Grid Data Function	7.00	1.25	0.89
Update Entity Graphics Function	23.90	5.09	3.64
Change State Function (End) (a and b)	25.90	8.98	6.42
a. Unload CDpp	25.30	8.46	6.05
b. Unload Application	0.60	0.52	0.37
Unload CDpp (composed of a and b)	45.00	14.17	10.12
a. Clear Stored Entities Function	23.20	7.21	5.15
b. Remove All Graphics Function	21.80	6.96	4.97

Table 1: UI Thread Function Timing

In examining the performance data of the function operations and to allow modification of code specifically to discrete event simulations, the JavaScript code was not compressed as it inhibits the developer's understanding and debugging ability by renaming variables. With this in mind, the timing evaluation of the functions of an entity's lifecycle from parsing to deletion on the UI thread was examined and is presented in Table 1.

Unlike other languages, JavaScript's pass-by-reference only allows one owner at a time. Though it is faster to pass data as a reference, evaluation of the largest Cell-DEVS simulation results file demonstrated that passing the uncompressed data between threads by value took 1600 milliseconds generating a thread communication speed of 17000 kB/s. The uncompressed data size was 28 MBytes and was determined by a developed JavaScript function similar to C's sizeof function.

Figure 7: Data Grid View of the Thin Client's Generated Entities

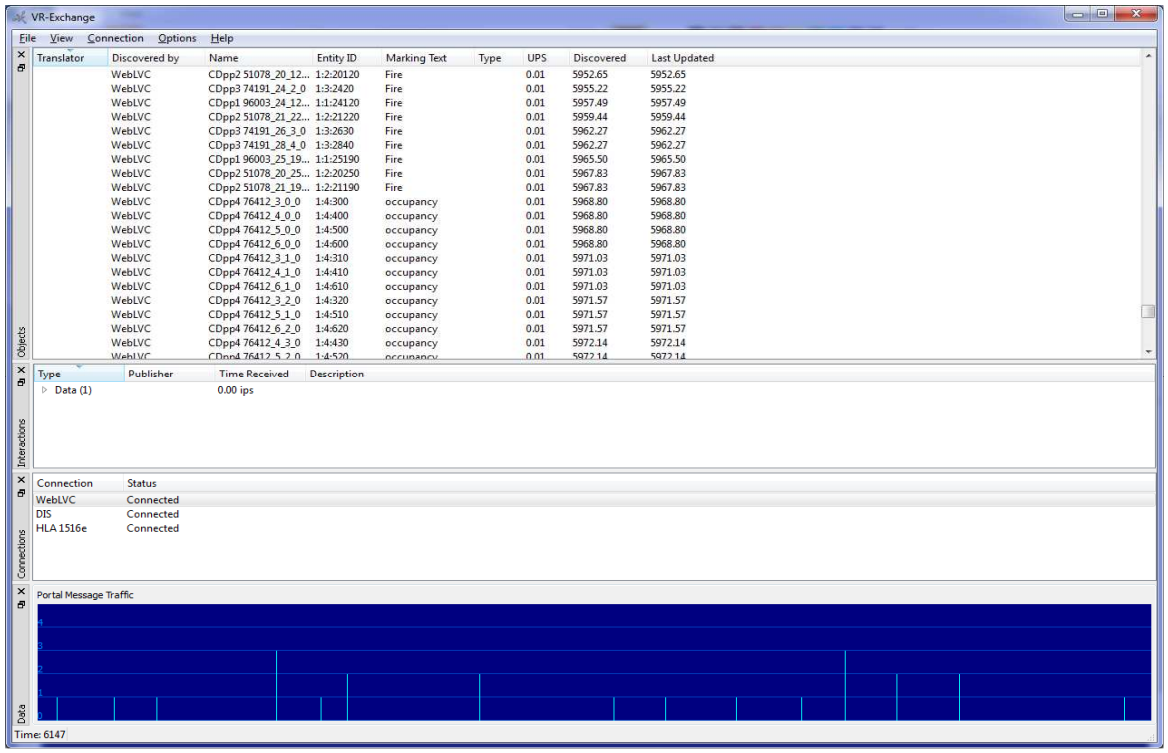

Figure 8: WebLVC Server Entity Population

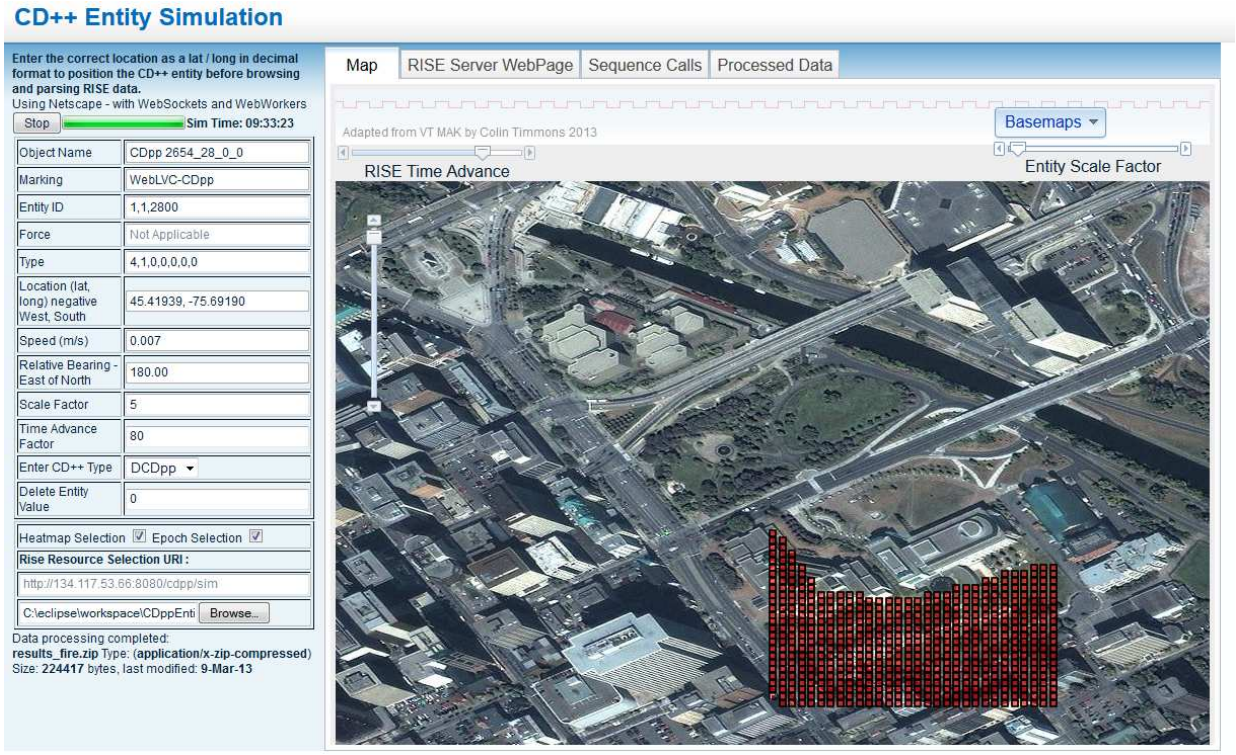

Figure 9: WebLVC Server Entity Population

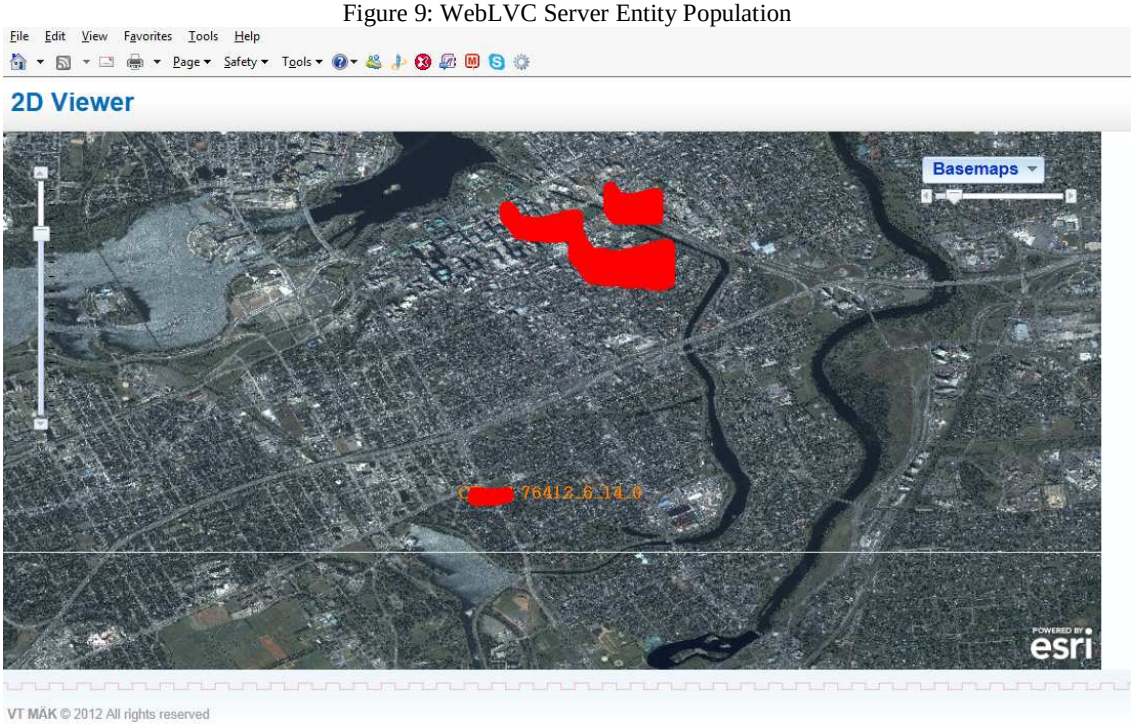

Figure 10: Verification of Multiple Clients With Different Discrete Event Models

Because of the amount of entities generated, a data view pane was developed to allow the user to inspect the Cell-DEVS grid cells. The pane displays the textual file line number after all duplicates are eliminated, the simulation time since the thin client was loaded, the discrete event simulation time, the graphic layer, its Cell-DEVS population value, its latitude and longitude position, and the bearing and distance from the Cell-DEVS grid origin. This is illustrated in Figure 7.

Using multiple thin clients to populate geographic entities, examination can be performed on the WebLVC server's acceptance of the data. This is confirmed in

Figure 8 and shows the recording of the entities to be passed into a distributed simulation from the multiple thin clients.

Figure 9 illustrates the HTML5 webpage and the entity generation of the developed thin client while multiple discrete event simulations are illustrated in Figure 10. The verification of the multiple discrete event models in a distributed simulation in Figure 10 uses MÄk's 2D viewer.

3.9. Conclusion and Future Work

In this paper, we introduce a technique for injecting discrete event simulations into traditional distributed simulations. Through new technologies and standards, a thin client was used to access, extract, manipulate, calculate and format the discrete event simulation data located on the RISE. This data was formatted into a structure acceptable for bridging web based simulations and distributed simulations via MÄk's WebLVC server.

The thin client technique was successful in accessing and manipulating the different service types of models and their respective simulation executions in order to generate entities based on the models and develop the entity's parameters that would be of significance to a distributed simulation. Multiple discrete event simulations were independently injected into the distributed simulation and monitored on a separate platform.

The WebSocket technology of HTML5 and the new semantic and syntactic elements of HTML5 combine to generate an operating platform that is equal to the bi-directional communication channels and visual representations employed in distributed simulations. The HTML5 multi-threading capability and the File API deserve credit as they promote the thin client to the level of traditional modern language applications that have successfully employed these technologies.

No longer is there a clear distinction between the distributed simulations and high performance of new HTML5 web-based simulations as WebLVC protocol attempts to become the standard interoperability protocol to adequately link new web-based applications with each other and with traditional distributed simulations.

Future work in this technique of the thin client can take many different paths. Firstly, with respect to this technique, once the 3D CSS becomes standardized and thus portable across thin clients, the orientation of the discrete event simulation can be re-calculated based on the location and orientation of structures on the ESRI ARCGIS map. With respect to the capabilities of the WebLVC protocol, the mapping of the HLA-to-WebLVC and WebLVC-to-HLA must first be accomplished and then the discrete event models can be injected into a HLA-only distributed simulation as a federate without the requirement of being FOM specific.

Another area of interest is to apply the technique to inject other simulation types into the distributed simulation. Conversely, traditional distributed simulation data can now be exported into web-based simulations. This would be advantageous for new distributed simulations attempting to deliver capabilities via the Service Oriented Architecture (SOA) and the user's selection of the structured collections of discrete modules that would be available.

REFERENCES

Al-Zoubi, K., Wainer, G., 2009. Using REST Web-Services Architecture for Distributed Simulation. *23rd Workshop on Principles of Advanced and Distributed Simulation*. pp. 114-121. June 22-25, Lake Placid, NY, USA.

Ameghino, J., Troccoli, A., Wainer, G., 2001. Models of complex physical systems using Cell-DEVS. *Proceedings of the 34th Annual Simulation Symposium*. pp. 266-273. April 26, Seattle, WA, USA.

Berners-Lee, T, 1996. *Universal Resource Identifiers – Axioms of Web Architecture*. World Wide Web Consortium. Available from: http://www.w3.org/DesignIssues/Axioms.html [accessed 6 October 2012]

Bizub, W., Bryan, D., Harvey, E., 2006. The Joint Live Virtual Constructive Data Translator Framework – Interoperability for a Seamless Joint Training Environment. *Meeting Proceedings RTO-MP-MSG-045*, pp. 9-1-9-8. September 1. Neuilly-sur-Seine, France.

Fielding, R.T., 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Thesis (PhD). University of California.

Hollenbach, J.W., 2009. *Inconsistency, Neglect, and Confusion; A Historical Review of DoD Distributed Simulation Architecture Policies.* Available from: http://www.sisostds.org/DesktopModules/Bring2mind/DMX/Download.aspx?Command=Core_Download&EntryId=28991&PortalId=0&TabId=105 [accessed 8 August 2013]

Hickson, I., 2012. *Web Workers. W3C Working Draft 01May 2012*. World Wide Web Consortium. Available from: http://www.w3.org/TR/workers/ [accessed 29 Devember 2012]

Lubbers, P.; Greco, F., 2009. *HTML5 Web Sockets: A Quantum Leap in Scalability for the Web.* Available from: http://www.websocket.org/quantum.html [accessed 29 December 2012]

MÄk, 2012. *WebLVC*. Cambridge, MA, USA. Available from: http://www.mak.com/weblvc/. [accessed 15 October 2012]

NATO, 2013. *NATO M&S Glossary Version 0.3*, North Atlantic Treaty Organization (NATO) Modeling

and Simulation Subgroup (MS3) of the NATO M&S Group (NMSG).

Pieters, S., 2013. *Differences from HTML4. W3C Working Draft 28 May 2013*. World Wide Web Consortium. Available from: http://www.w3.org/TR/html5-diff/ [accessed 9 August 2013]

Ranganathan, A., Sicking J., 2012. *File API. W3C Working Draft 25 October 2012.* World Wide Web Consortium. Available from: http://www.w3.org/TR/FileAPI/ [accessed 28 December 2012]

SISO, 2007. *Distributed Interactive Simulation Product Development Group (SISO), Guide for: DIS Plain and Simple. SISO-REF-020-2007*. Available from: http://www.sisostds.org/DesktopModules/Bring2mind/DMX/Download.aspx?Command=Core_Download&EntryId=29302&PortalId=0&TabId=105 [accessed 15 October 2012].

Smith, K. 1995. *Distributed Interactive Simulation Network Manager. ARL-TR-780*. Army Research Laboratory. USA. Available from: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA298053 [accessed 28 December 2012]

Wilson, A., Weatherly, R., 1994. "The Aggregate Level Simulation Protocol: An Evolving System". *Proceedings of the 1994 Simulation Conference* pp. 781-787. Orlando, FL, USA.

AUTHORS BIOGRAPHY

Colin Timmons is an Aerospace Engineering Officer in the Canadian Armed Forces and is presently employed in the Department of National Defence / Canadian Armed Forces (DND/CAF) Synthetic Environment Coordination Office (SECO). Responsible for the standards and architecture of M&S in defence, he maintains national and international interoperability in joint and combined force generation. Colin Timmons has over 25 years of experience with computer hardware, computer software and systems engineering design practices, project management and system implementations. Presently, he is completing his MASc in Modeling and Simulation at Carleton University in Ottawa, ON, Canada.

Gabriel A. Wainer (SMSCS, SMIEEE) received the M.Sc. (1993) and Ph.D. degrees (1998, *with highest honors*) at the University of Buenos Aires (UBA), Argentina, and Université d'Aix-Marseille III, France. In July 2000, he joined the Department of Systems and Computer Engineering at Carleton University, where he is now Full Professor. He has been a visiting scholar at ACIMS (The University of Arizona); LSIS (CNRS), and INRIA (Sophia-Antipolis), France. He has been invited professor at the UCM, UPC (Spain), Université Paul Cézanne, Université de Nice (France). He is the author of three books and over 270 research papers; he edited nine other books, and was a PC member/organizer of over 120 conferences, being one of the founders of SIMUTools, SimAUD and the Symposium of Theory of Modeling and Simulation. He has been appointed as Program Chair of the Winter Simulation Conference in 2017. Prof. Wainer is the VP Conferences, Was VP Publications, and a member of the Board of Directors of the SCS. He is Special Issues Editor of SIMULATION, member of the Editorial Board of IEEE Computing in Science and Engineering, Wireless Networks (Elsevier), and Journal of Defense Modeling and Simulation. He has been the recipient of various awards, including the IBM Eclipse Innovation Award, SCS Leadership Award, and various Best Paper awards. He has been awarded Carleton University's Research Achievement Award (2005-2006), the First Bernard P. Zeigler DEVS Modeling and Simulation Award, the SCS Outstanding Professional Award (2011), Carleton University's Mentorship Award (2013) and the SCS Distinguished Professional Award (2013). His e-mail and web addresses are <gwainer@sce.carleton.ca> and <www.sce.carleton.ca/faculty/wainer>.