Control and Management of Clustered Simulation Systems

Edwin Michael Bearss
Trideum Corporation
Huntsville, AL
mbearss@trideum.com

Roy F. Zinser
Trideum Corporation
Huntsville, AL
rzinser@trideum.com

Keywords:
Configuration Management
Risk Management
Networking
Cloud Computing

ABSTRACT: *The purpose of this paper is to discuss the common issues experienced during configuration and monitoring of systems in a distributed modeling and simulation (M&S) exercise. Distributed M&S exercises often require a variety of diverse applications to interoperate over a network that is not controlled by a single entity. As these applications are often created by different vendors, there is currently not a standard in place to manage and monitor each of these systems. The lack of management ability increases the time and cost of configuring these systems and can pose a threat to the reliability of the exercise overall.*

Due to the sensitive nature of exchanged data, security can also be a concern with these distributed exercises. Military M&S exercises typically require robust firewalls and host-based security solutions to be in place. These systems make it difficult to deploy commercial remote management tools to address this problem. In addition to these security concerns, this software is usually developed for system administrators and is not well tailored for M&S exercises.

This paper discusses our lessons learned while iterating over multiple applications developed while addressing the problem of cluster management. While the solution we developed served the purpose of managing our infrastructure, we feel that developing a standard would greatly assist other teams facing similar configuration issues. The issues covered in this paper are most prevalent in the military M&S community but will likely occur in other environments where security is a concern.

1. Introduction

Many current simulation exercises run in a localized configuration with multiple computers connected over Local Area Networks. Each of these machines run their own operating system and copy of the simulation software. Distributed events connect these localized clusters via Wide Area Networks. This clustered nature of computing systems along with the security requirements of working with sensitive data often pose issues for configuration management and initialization of the simulation exercise. Poor configuration management could compromise the integrity of the mission or cause spillage of sensitive information. The consolidation of data centers and push to host legacy simulations in cloud environments further complicates the control and management of simulation applications.

Simulation vendors assume expertise by system administrators and technicians to not only install and configure, but also to instantiate and run their products. Running commands manually on each machine to launch simulation software is tedious and error prone, especially for non-technical operators. Typically, event sites develop ad-hoc scripting solutions to launch their simulations. Most use a mix of command line interface, simple GUI launching scripts, or custom shell scripts running over a remote shell protocol. Each of these methods have clear disadvantages. Custom developed scripts are usually plagued with security vulnerabilities and require constant maintenance to keep updated.

Investing in a commercial solution could be a viable alternative to these locally developed scripts, however, at the time of writing we have been unable to find a solution that is flexible enough to support the myriad of M&S tools required for complex distributed simulation environments. Commercial lab management tools typically only support launching of the base operating system and are a poor fit for the military simulation environment. Many of these off-the-shelf tools require installing agent software on each computer, which complicates the Risk Management Framework (RMF) and Authority To Operate (ATO) information assurance posture. These tools also don't provide a user interface that is well tailored to the layout of typical simulation exercises.

In order to develop an effective solution to this problem, we created an application leveraging modern web technologies to connect to clients and provide an intuitive user experience. This new configuration management and monitoring mechanism proved effective at supporting both local and distributed simulation events. Establishing a standard to interface with simulations could vastly improve the capability for practitioners to launch and control their systems.

2. Understanding the Problem

Due to the nature of simulation software, many simulation applications use rudimentary user interfaces, or operate headless without a GUI at all and are required to be run using shell commands. This lack of a friendly user interface makes these tools difficult to use for typical simulation operators and more time consuming to start the software on multiple computers. These issues are often mitigated by creating startup scripts that run a set of commands that start the applications. However, these are usually inflexible and require a technically skilled person to develop. Some flexibility can be provided by these scripts through user input, or reading system configuration or files, but these quickly grow in complexity making them a non-scalable solution.

Configuration management is a common issue present when dealing with multiple systems. Ensuring systems have a common software baseline, while allowing for quick patching can be a difficult task. As the number of systems and variations of software grows, this becomes a huge concern, as a single mis-versioned system can cause anomalies in the simulation event. Most standalone systems also have no ability to monitor the health of the current simulation, requiring the user to infer the current status from logs and system resource utilization.

Even advanced simulation systems, such as One Simi-Automated Forces (OneSAF), generally have poor monitoring systems in comparison to modern standards. These tools are usually not customizable and are often not web based, requiring a user monitoring the system to be locally present at a particular computer.

Simulation data collected post-event can also pose some difficulty for operators. This usually involves collecting the data files and logs created on each machine and sending them to a single location for analysis. Using a tool can ease this collection, or even allow for some run time analysis to be performed during execution.

Providing the technical support for multiple distributed simulation sites and Federation Control for the Battle Lab Collaborative Simulation Environment (BLCSE) our team faced all the above challenges. Large events routinely included several hundred simulation systems at multiple distributed sites. Deploying, running, monitoring, and consolidating data across the environment was a persistent friction point.

3. Architecture Issues

Many real-world simulation events rely on different simulation systems, each of which are sometimes systems-of-systems. This presents issues when controlling and monitoring simulation events. It is common for events to require systems from multiple different vendors, making it difficult or infeasible for the practitioner to make efficient software to control the entirety of a simulation event.

Military simulation events are often composed of various geographically dispersed locations with hardware and software resources under the control of individual sites. This separation, in both location and management, often prevents systems from running a common baseline. The ownership of these resources, and the requirement to meet security guidelines also makes these sites resistant to adopt management software not directly controlled by the site itself.

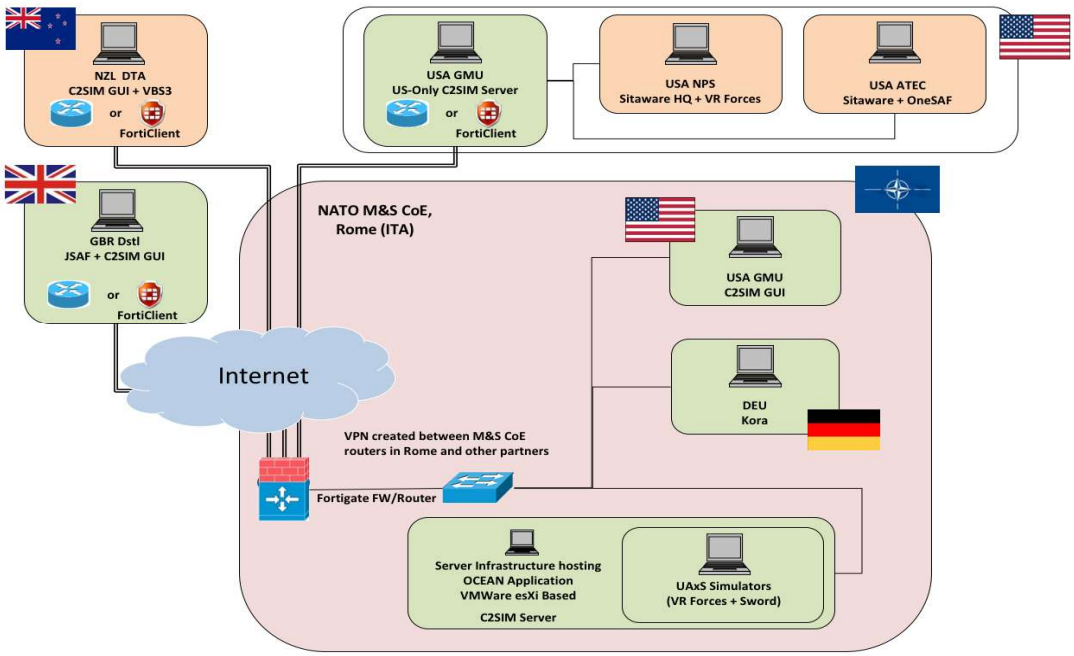

Figure 1: Coalition Warrior Interoperability eXercise (CWIX) 2019 Network Architecture (Bearss)

4. Commercial Tools

There are two major categories of commercial tools that are suitable to manage a group of simulation systems. Computer lab management software, such the open-source software Veyon, are often used by school administrators to monitor students and maintain configuration on lab computers. These systems could be used to control a simulation event as they have the ability to monitor processes running on a system and even allow remote view/control of the client systems. These systems are often user friendly and provide strong configuration management capability, allowing a single person to patch and update connected systems. However, these lab management systems are not tailored to simulations, and would not have user interfaces tailored to a simulation exercise. These lab monitoring systems also require extremely invasive user agents that

would be difficult to operate in a secure environment. Figure 2 shows a typical lab management interface, with the option to monitor and control each computer connected to the system.

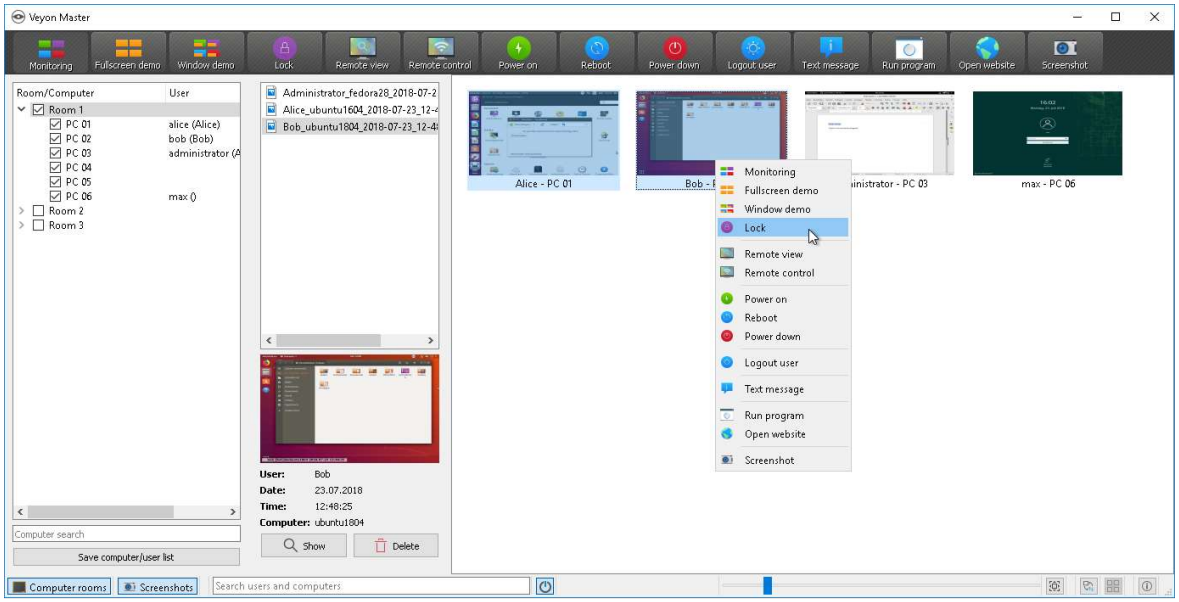

Figure 2: Veyon Management Software

Another potential solution is to use a job automation system, such as Ansible or Rundeck. Both tools use preconfigured "playbooks" or "runbooks," to run jobs on connected systems. Unlike the lab management software that relies on an agent program, these IT automation systems typically rely on remote shells with stored credentials to carry out their functionality. These systems are very capable and can be used to patch systems, start simulation events, and aggregate data after the event is over. The major disadvantage of these systems is the complexity and difficulty to configure properly, especially in non-static environments. Figure 3 shows the progress of an Ansible playbook being run on Ansible Tower with the overall status on the left and task details on the right.

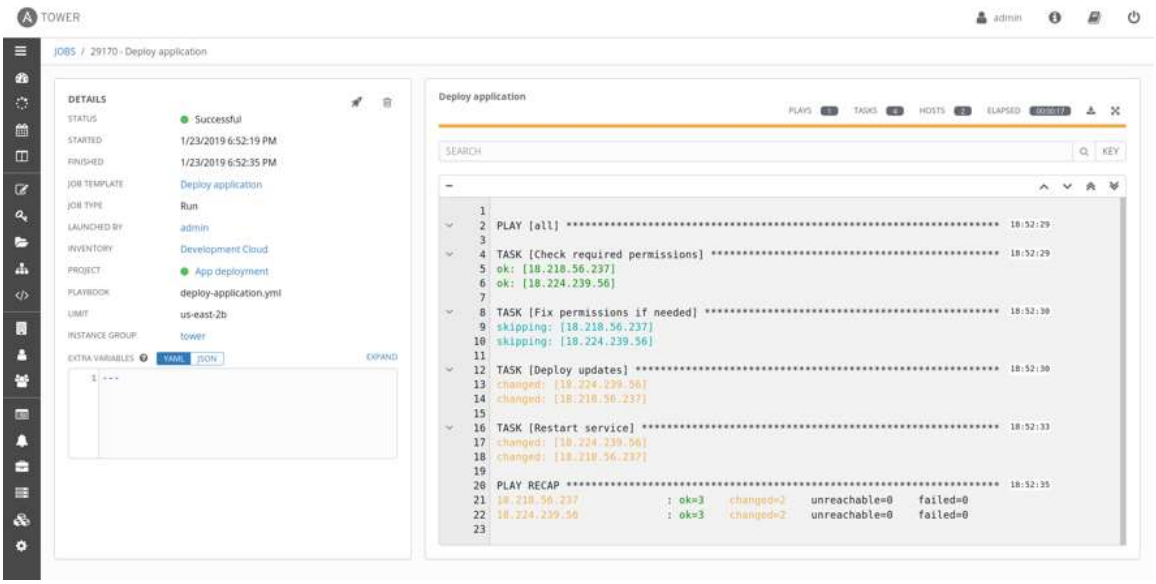

Figure 3: Ansible Tower

5. Developing a Solution

While attempting to automate portions of our workflow, we iteratively developed various tools with increasing complexity and user friendliness. Starting with a simple shell scripts to launch multiple systems, a simple text-based GUI was then added to make it possible for operators to run the commands. Next a desktop solution with a small Java based agent on each controlled machine and a Java based control application run on a single machine was developed. These applications communicated through encrypted java network sockets. This solution provided a much better user interface, however, there were a few major disadvantages. The controller application was only available from a single machine and required SSH keys for each client to be stored locally making the control system non-portable. The Java agent was also required to dispatch any system commands through the shell, making it difficult to monitor and control jobs.

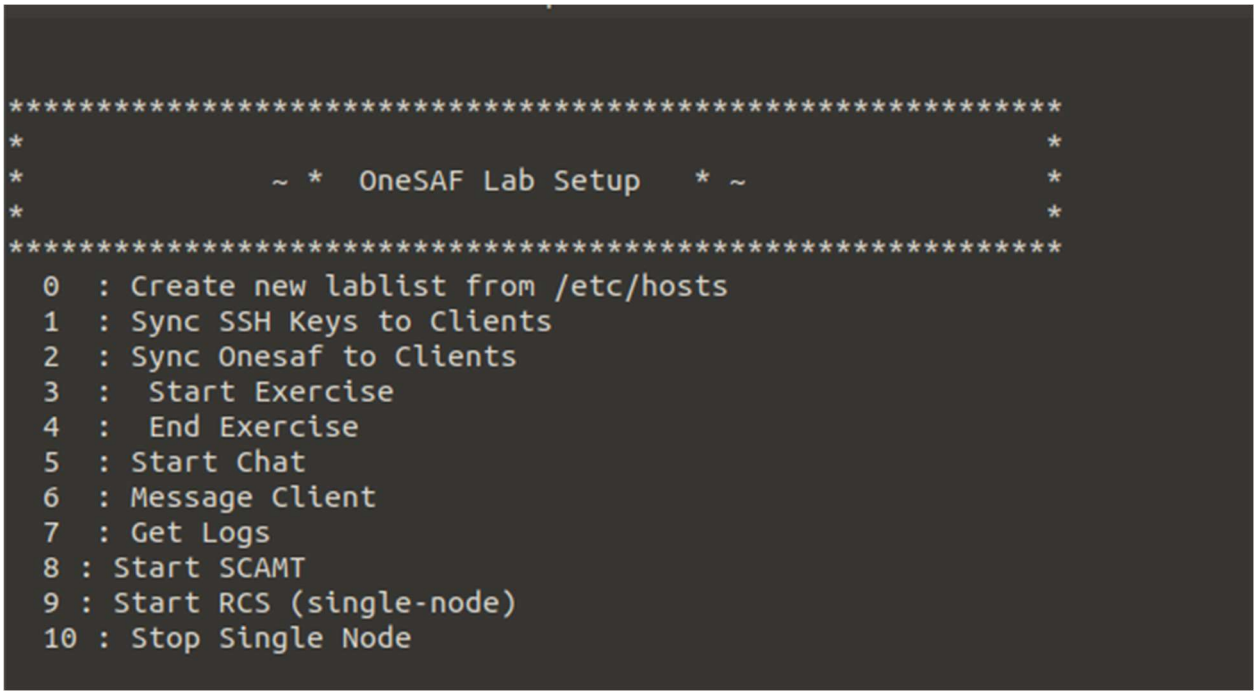

Figure 4: Early OneSAF lab manager scripts

Our next iteration leveraged Ansible to provide configuration management capabilities and update the simulation software with a separate web-based application to support simulation controls. Our team developed a series of Ansible playbook templates that local administrators could easily modify, and source code configuration enabled by a local Git repository. These playbooks, run on a biweekly basis, distributed the most up to date version of each simulation tool in use. Ansible was chosen for its fault-tolerant nature and being run by system administrators, a GUI was not a requirement.

To allow operators to control simulation events via a web-browser, we developed a web-based application called the OneSAF Web Launcher (OWL). Our solution uses a NodeJS application that communications to Python based clients using websockets. Each client runs a small agent program responsible for executing commands and reporting system statistics back to the controller application. The NodeJS application is responsible for aggregating these statistics and providing a web interface showing the status of each machine and allowing the user to start various simulation software.

The NodeJS application also provides a REST API that can be used by simulation systems to provide additional information, such as OneSAF performance metrics. In addition to the Python agent software, a new OneSAF extension was created to periodically send performance metrics, such as simulation load and the time discrepancy between machines, to the controller software. OneSAF was chosen as a first candidate for application-level monitoring due to its extensibility and ease of development. OneSAF already contains a deprecated REST performance monitor, so modifying it for our purpose greatly reduced the complexity of the task.

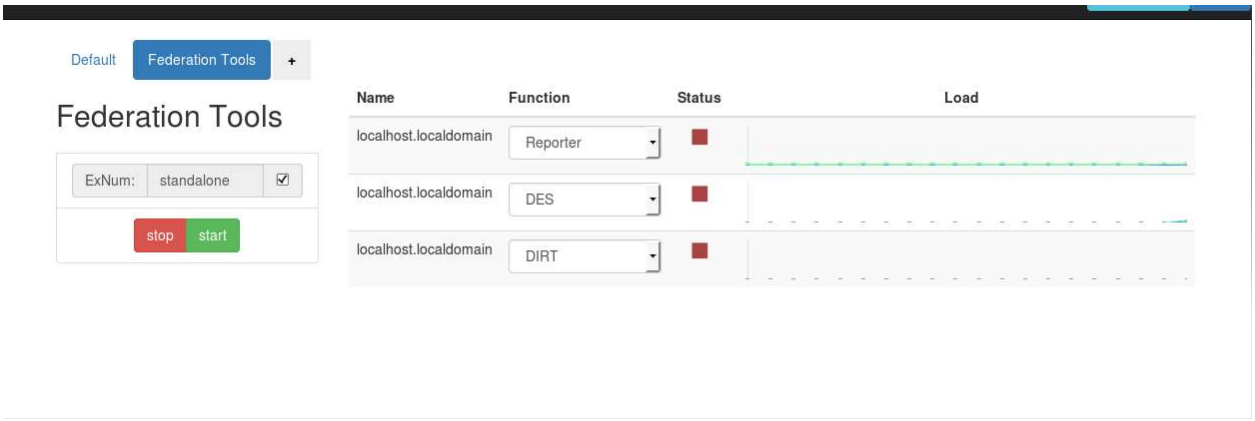

Figure 5: OneSAF Web Launcher (OWL)

We found using web-based technologies to be the biggest improvement during our development efforts. The ability to access the dashboard from any machine, along with the security provided by TLS certificates were huge considerations in our decision to move to the NodeJS solution. Although only used for basic OneSAF statistics, the REST API provided by the application showed huge potential. Expanding this API could provide countless statistics that would be unavailable with most generic commercial management tools, such as application specific health status, and even events occurring within the simulation.

In addition to developing the software application, performing the actions necessary to comply with the Risk Management Framework (RMF) was a major endeavor. Both Python and NodeJS rely on package managers, PIP and NPM respectively, to import libraries from third party resources. This allows for easy importing of libraries to perform security related functions (i.e., TLS negation, session tracking, logging, etc.), but requires each library to be approved. After obtaining this list of dependencies, gaining the approval from the Configuration Control Board was straightforward. After approval, the developed software was included in the list of simulation applications for the event Operating System Secure Host Baselines.

6. Related Work

PEO STRI is currently developing a publish/subscribe alternative to HLA named Bifrost. This new protocol runs on modern web technology on a microservice architecture. This update to a more modern topology greatly reduces network utilization and has the ability for participating systems to use a common high-level interface to exchange information. This makes it possible to perform complex functions though an external interface. These actions can be simulation specific, such as generating behaviors for entities, or be used to control the exercise by starting or stopping application instances or collecting data. Although still early in development, this new interoperability protocol could have a huge impact on future distributed simulations.

7. Future Work

We aim to increase the ability of the controller application to communicate with the simulation software directly, rather than through an agent application. Many simulating systems allow for extensibility, allowing easy addition of software to their codebases. Allowing the controller and simulation system to talk directly increases the ability to monitor the application and reduces the complexity of the agent software on the machine. Ideally, the agent would be no longer required at all. Moving to a microservices like architecture, such as Bifrost, could allow an entire distributed exercise to be created as needed without relying on any sort of custom host-based agent application.

8. Conclusion

System configuration and monitoring continues to be an issue in the military modeling and simulation environment. The requirement to quickly reconfigure software and the increased security requirements make many commercially available tools non-viable. In addition, the geographic distance and control of resources further complicate the problem. To effectively solve this issue, we believe a move to a more modern topology will be required. Implementing agent software on each machine that is custom tailored to each simulation is not a scalable solution. Developing a common control and monitoring protocol implemented by each simulation system could be a viable option. Developing and implementing a new standard will require effort, but we believe there will be a huge benefit to modernize the existing approach.

9. References

"Ansible Tower." *Ansible Tower | Ansible.com*, www.ansible.com/products/tower.

"Cross-Platform Computer Monitoring and Classroom Management." *Veyon*, Veyon Solutions, 26 Apr. 2019, veyon.io/.

Bearss, Edwin. "Development of a C2SIM interface prototype in OneSAF." Simulation Interoperability Workgroup 2020

Author Biographies

EDWIN MICHAEL BEARSS is a senior software engineer at Trideum corporation with more than 6 years of experience in Modeling and Simulation. He is also a current PhD student in pursuit of a degree in Computer Science with a focus in Modeling and Simulation at the University of Alabama in Huntsville.

Roy F ZINSER III is a M&S Integration Lead and Simulation-Mission Command Interoperability Architect at Trideum Corporation with more than 15 years' experience in Modeling and Simulation (M&S) and Mission Command Information Systems. He has supported M&S in multiple domains including Training, Experimentation, and Test & Evaluation.