A Simple Distributed Simulation Architecture for Emergency Response Exercises

Dennis McGrath, Amy Hunt, and Marion Bates
Institute for Security Technology Studies, Dartmouth College
dmcgrath, ahunt, mbates @ ists.dartmouth.edu

Abstract

This paper describes a simple distributed simulation for support of emergency response exercises. The simulation, called the Immersive Synthetic Environment for Exercises (ISEE), was inspired by the simplicity of web-based role playing games (RPGs) which have all the elements of distributed simulations, but do not rely on complex interoperability software or high-end graphics engines. ISEE uses PHP to generate user interfaces, MySQL as simulation middleware, and agent-based simulations written in Python determine the behavior and state of simulated objects. It uses a web-interface that allows players to see events, communicate, and assign resources to critical emergency response functions. Test showed that up to 1000 entities could be represented with a quarter second simulation time step. The application was deployed as a prototype to support a mass-casualty emergency response exercises in May of 2005. Results of the exercise showed that the intended value of distributed simulation (multi-user, immersive experience) was achieved using simple tools with a rapid development time.

1. Introduction

The early success of the distributed interactive simulation (DIS) protocol in enabling distributed simulation [Clarke] has been overshadowed in recent years by the increasing complexity of interoperability protocols such as the High Level Architecture (HLA). The increasing complexity and associated cost of distributed simulation are acting as barriers to development, and consequently as barriers to widespread use of distributed simulation outside the military domain. Simulation interoperability remains a worthwhile pursuit, but bringing distributed simulation to the masses may require simpler solutions.

The Institute for Security Technology Studies (ISTS) at Dartmouth College is researching ways to improve the readiness of emergency responders[1]. One of the projects funded by the Emergency Readiness and Response Center (ER3C) within the Institute is applying synthetic environment technology to the emergency response domain with the goal of finding low-cost, effective methods for improving readiness of emergency responders at all levels. Specifically, the research seeks to apply distributed simulation technology and game engines to create synthetic environments that represent the specific challenges of emergency response.

Among the products of this research is a web-based simulation called the Immersive Synthetic Environment for Exercises (ISEE). ISEE began with the goal of applying military simulation and/or commercial game technology to create a real-time, multi-user, immersive exercise framework. Surprisingly, however, the specific needs of the emergency response community led ISEE development away from the established protocols and tools of military distributed simulation. While the arrow of progress in technology development often pushes toward more complicated solutions, sometimes the needs of the end users mean that "less is more".

Inspired ultimately by web-based role playing games, ISEE uses PHP, MySQL, and Python for user interface, middleware, and simulation engine respectively. Although based on simple tools, the end result has the characteristics of a collaborative, synthetic environment, developed in a matter of weeks.

[1] Supported under Award number 2000-DT-CX-K001 from the U.S. Department of Homeland Security, Science and Technology Directorate. Points of view in this document are those of the author(s) and do not necessarily represent the official position of the U.S. Department of Homeland Security or the Science and Technology Directorate.

2. Background

We live in an age of acute awareness of the potential for catastrophic events. The most effective way to prepare for a response and recovery for these events at the local or regional level is through emergency response exercises. Traditionally, these exercises have been either tabletop exercises (TTX) with limited realism or full-scale (FSX) exercises requiring the costly deployment of vehicles, personnel, and equipment.

Because of cost limitations, emergency response tabletop exercises are held more frequently than full-scale exercises, but the TTX paradigm suffers from several weaknesses. While an exercise moderator is responsible for controlling the flow of events, players in tabletop exercises are often asked to contribute to the narrative of events, and players naturally tend to minimize problems and maximize the effectiveness of their own actions. As a result, players often commit more resources than they actually have available, and estimate unrealistic timelines for the deployment of those resources (eg. "My HAZMAT unit would be on scene in five minutes"). Another challenge in the TTX format is effective capture of events and communications for after action review. Typically "scribes" are assigned to the task of taking notes which may vary greatly in detail and accuracy.

Those familiar with distributed simulation, which has been used extensively in defense applications, can see the obvious benefit that can be derived from the applications of distributed simulation technology to emergency response exercises. Improved realism and more effective after action review are among the benefits of a computer-simulated scenario. In fact, several efforts have been aimed specifically at reworking military simulation frameworks for civilian use [Smith] [Kincaid]. These "swords to plowshares" efforts are based on several assumptions:

- The emergency response community is similar to the military community as a market for simulation technology
- Existing models are appropriate and available for re-use by the emergency response community
- The drivers for military simulation (numbers of entities, time management) are similar to those for emergency response simulation

But use of distributed simulation for emergency response exercises is currently rare. With the exception of a few technology demonstration projects, emergency responders have not incorporated distributed simulation into their training and exercise paradigms. This is due in large part to the fact that the assumptions about an easy transfer of simulation technology from military domain to homeland security are wrong.

The reality of technologies for emergency response is that simplicity, ease of use, and cost trump all other concerns. Any tool that is to be used frequently by emergency responders must be flexible enough to represent different locations, scenarios, and community resources. Simulations for emergency response exercises must be generated quickly, run on ordinary office computers, and they cannot involve expensive licensing agreements.

Commercial game engines have emerged as an alternative to simulation frameworks in recent years, and generally have the necessary elements of synthetic environments (immersive, real-time, distributed). Ease of use and low cost make game-based simulation an attractive option for simulation development [McDowell], and several research efforts in recent years produced game-based training environments for first responders. These training systems are visually rich and immersive, but the cost of graphically-intensive game development is still very high, given the resources required to generate realistic environments and characters. Furthermore, tabletop exercises involve incident command-level responders, who generally do not experience the event from first-hand visual cues. Instead, these players build situational awareness from situation reports from the field. Consequently, an immersive simulation for command-level users is not necessarily a graphical simulation.

Given this brief survey of distributed simulation technology, game engines, and the needs and limitations of the emergency response community, the requirements for a distributed simulation framework to enhance the realism of tabletop exercises include:

- Real-time
- Multi-user
- Low cost, no licensing fees
- Able to run on low-end computers with limited bandwidth connections
- Simple user interfaces and installation
- Flexible enough to support multiple scenarios

3. ISEE Architecture

The ideal framework for a distributed emergency response simulation based on these requirements is an open simulation or game engine with minimal graphical and network overhead. The search for such a framework led to a specific sub-class of games—the turn-based online role-playing game – which acted as inspiration for ISEE. Role playing games (RPGs)

generally involve fantasy-based characters such as wizards, dwarfs, and elves collectively fighting against monsters with some combination of weapons and spells. Taken in the abstract, emergency response involves different kinds of characters collectively applying resources (ambulances, fire engines, rescue gear, etc.) to solve problems (treatment of casualties, fire suppression, etc). The analogy is spelled out more thoroughly in Table 1.

	RPG	**Emergency Response**
Players	Wizards, fairies, elves, etc.	Fire, EMS, Police, Incident command
Inventory	Weapons, shields, magic spells	Fire engines, Ambulances, Police cruisers
Challenges	Monsters	Fire suppression, casualty management, public safety
Communication	Instant messaging and player forum	Radios, cell phones, pagers

Table 1 – Role playing games vs. emergency response exercises

Several web-based RPG's allow users to play through any web browser without the installation of special software. The simplicity of web-based RPGs inspired a simple architecture for ISEE which retains the fundamental components of distributed simulation (user interface, interoperability middleware, simulation engine) while satisfying the cost and simplicity needs of the emergency response community. The result is an immersive, distributed simulation, developed in several weeks which allows incident commanders to rehearse emergency response procedures in real-time, with very little network bandwidth or graphical horsepower.

Figure 1 shows the three-tiered structure of ISEE, with html-based user interfaces generated by PHP, a MySQL middle layer, and simulation agents and objects written in Python. MySQL is an open database management system, which in this application is essentially used as middleware. It keeps track of persistent simulation object state and saves all events and communications for after action review. PHP is an open server-side scripting language for generating web content which has strong hooks to MySQL. PHP proved to be a good tool for customizing individual interfaces for different player types. Python is an object-oriented scripting language which likewise works well with MySQL databases. Python does not have the computational performance of Java or C++, but its simplicity makes it well suited for rapid simulation development.

Python agents and simulations can connect to the database either locally (on the server) or remotely (across the network). Object state can be modified by users through the web interface and by the python agents. For instance, a user might assign a particular unit such as an emergency medical service (EMS) team to one of several task areas, including triage, treatment, or transport of casualties. The assignment (state) of the EMS unit is retrieved from the database by the EMS agent which simulates the behavior of the unit. When the agent calculates a new state for the unit, it publishes this new state to the database, where it can be pulled by the client through the web interface.

Likewise, the state of non-agent objects with persistent state is kept in the database. For instance, a fuel fire object will continue to grow according to the "standard curve" for petroleum fire dynamics until a player assigns a unit to fire suppression, at which point the fire dies down according to an extinguishing algorithm based on preburn time, flow rate of extinguishing foam, and maximum fire temperature. As the state of the fire changes, the new values are published to the database.

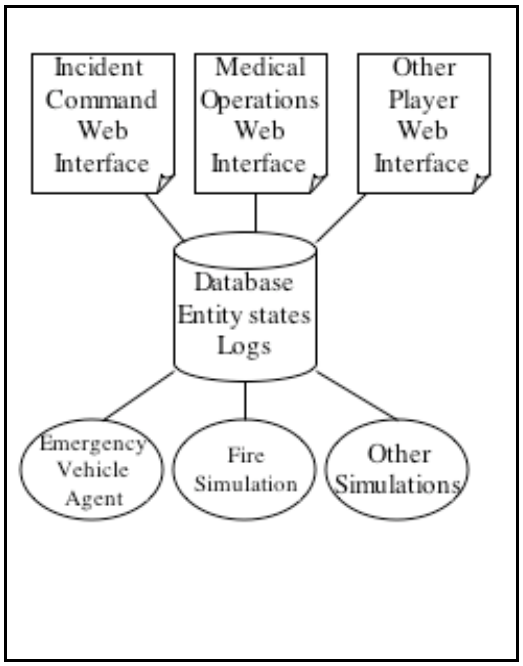

Figure 1 - ISEE architecture

4. User Interface

The ISEE user interface (Figure 2) is designed around three principles of incident management:

- Situational awareness
- Communication
- Resource management

Players log on as one of several IC roles (incident command, dispatch, fire, police, etc.) with a username and a password. The user is taken to a main page consisting of several windows and panels. Situational awareness comes primarily from the Event window, which notifies players of significant events that occur during the simulation. Additional situational awareness is derived from maps and images of the incident scene as well as weather conditions and an elapsed time clock. Radio communication is approximated through a text message panel, where users type messages and "send" them over one of several channels (police, fire, broadcast) which are logged for after action review.

Resources are shown and managed through a pop-up window which identifies all of the emergency vehicles on scene that have been assigned to that particular player by incident command. Resources include people, equipment, and vehicles that can be tasked to address specific problems (fire suppression, traffic control, triage, casualty treatment, etc.). Each player is equipped with a set of functions to help them complete their responsibilities (contact outside agencies, set up treatment areas, establish an incident command location, etc.) per the disaster response plan. Players are also given visual indicators that allows them to see how they are progressing against their specific challenges (fire, traffic, casualty management). These indicator bars get the current state of the challenge from periodic queries to the database.

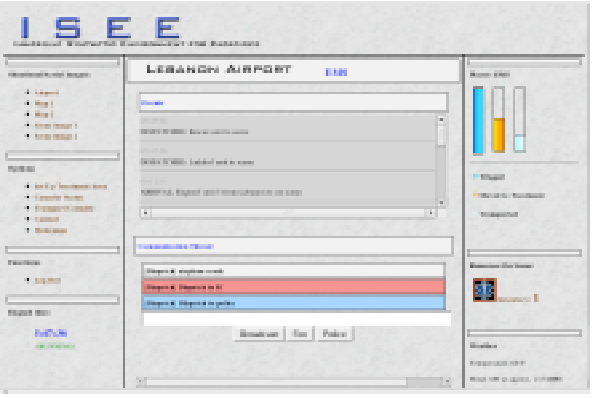

Figure 2 - Player interface

Some objects (resources) are owned by different players during the course of the simulation. Object ownership is therefore variable and stored as an object feature. For instance, an ambulance initially "belongs" to dispatch who sends it to the scene. When it arrives on scene, ownership is transferred to incident command. Incident command then has the option of assigning it to various subordinate branches (police, fire, medical). The branch commander then owns the asset and has the option of returning it to incident command or releasing it, which effectively transfers it back to dispatch. Users only have visibility and control over resources that they "own".

5. Performance Tests

Questions about performance with this server-centric simulation are valid concerns, given the overhead of database storage and the use of an interpreted (rather than compiled) simulation language. The centrally located database creates the possibility of a choke point. Since clients only query the database several times per minute, the larger concern is the interaction between simulations and the database, which may update state information several times per second. The critical question to be answered is how many entities can be simulated before the overhead of storage, network latency, and computation slow the update cycle to unacceptable levels.

Generally in the distributed simulation cycle, simulations must retrieve the current state of simulated objects, calculate a new state, and then publish the new state information to the middleware (or database in this case). This query->calculate->update cycle was the basis for performance tests. The scaleability of the ISEE architecture was tested to determine the time to update state values for simulations with up to 10,000 entities on a single processor server (Intel 1.7Gz) with 512 Mb of RAM.

Figures 3 and 4 show that times to query and update simulated entities using a central database as middleware. The results shown are for a server running both the MySQL database and the simulation on the same machine (i.e. connecting via localhost). The worst case shown for query and update is 10,000 entities with 16 features per entity, where the time to query and update were 125 and 250 milliseconds, respectively.

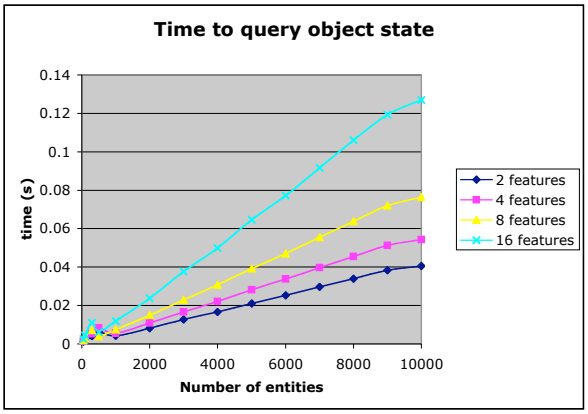

Figure 3 - Average time to query entity state for entities with 2, 4, 8, and 16 features (localhost)

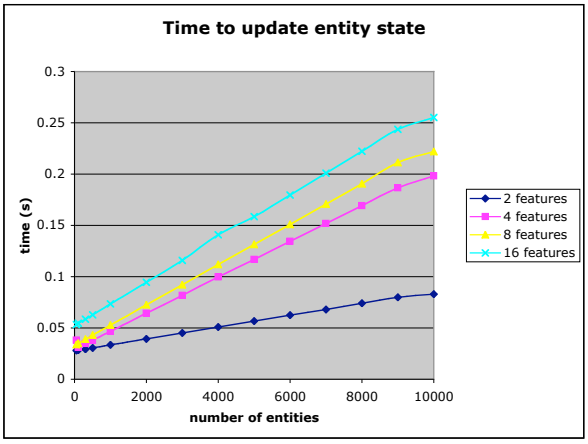

Figure 4 - Average time to update entity state for entities with 2, 4, 8, and 16 features (localhost)

Figure 5 shows the effects of network latency on the update cycle for entities with 16 attributes. Simulation on the local server increases the processor load, but does not suffer from network latency. While distributing the simulation to local network hosts alleviates some computational burden from the server, it adds more than 100% delay to the update cycle.

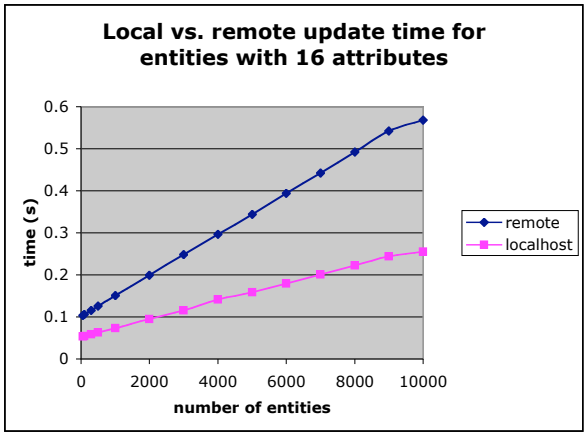

Figure 5 - Average time to update entities for distributed (remote) simulations and server-based (localhost) simulations

Figure 6 breaks down some of the components in simulation cycle for entities with 16 attributes. Individually, the time to query, calculate, and update (publish) entity state data are shown, along with network latency for remote simulation. The total (worst case) time is determined by the formula:

$$t_{total}=t_q+t_c+t_u+2(t_n)$$

where t_q is the time to query, t_c is the time to calculate new state, t_u is the time to update state information, and t_n is network latency, which is doubled because both query and update cycles experience network latency. This "worst case" study shows that with a .25 second time-step target, 1000 entities can be simulated with the ISEE architecture. Relaxing the update cycle requirement to .5 seconds allows approximately 4000 entities to be simulated without falling behind real time.

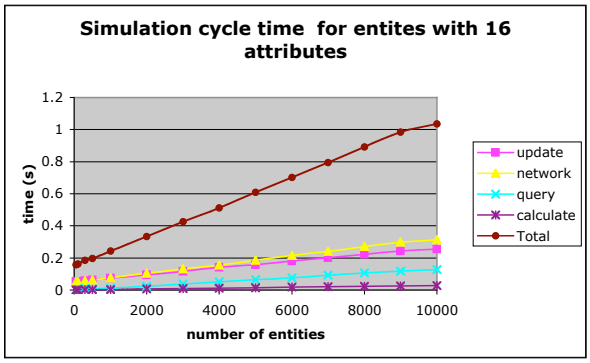

Figure 6 - Simulation cycle times for query, network, update, and state calculation

6. Prototype System

A prototype ISEE system supported a mass casualty exercise in Lebanon, NH in May 2005. The scenario involved an airport crash at a small regional airport which was mandated by the Federal Aviation Administration to maintain airport certification. The scenario involved an aircraft fuel fire and 20 casualties with injuries ranging from superficial to fatal. Players in the simulation included:

- Airport operations
- Dispatch
- Incident command
- Police
- Fire
- Emergency Medical Operations
- Hospital

Resources for the exercise were based on actual community vehicle inventories and capabilities. The exercise time was just under one hour from the report of the plane crash to dispatch until the transport of all living casualties from the scene. 21 local vehicles and 15 mutual aid vehicles (from neighboring communities) were called to respond. 188 significant simulation events were logged and 159 communication messages were recorded. Players reported that the level of interaction and realism far exceeded that of a traditional tabletop exercise. The comprehensive logging of all events also made an immediate after action review possible.

Negative feedback from users focused on the communication panel. Although similar to instant messaging (IM) or "chat" applications, users described the shift from audio cues to visual (reading) cues very confusing, particularly in real time. The motivation for using an IM function instead of permitting voice channels for communication (phone, radio), came from the assumption that capturing all communications was worth the relative inconvenience to the players. Based on the feedback, however, future research will be conducted into methods for capturing voice communications so that players will be able to communicate naturally.

7. Conclusions

The distributed simulation approach using MySQL as middleware architecture described here is clearly not the solution for all distributed simulation problems. It does not, for instance, support any sort of time management and is not able to support very large simulations (tens of thousands of entities) without suffering from performance issues. However, for small simulations (fewer than 1000 entities) and some middleware jobs such as entity state management, object ownership, and event logging, the approach has the advantage of simplicity without performance problems.

ISEE demonstrates that a simple distributed simulation based on open tools can be an effective method for improving tabletop exercises. Its lack of dependence on custom or proprietary tools, and its ability to run in a web browser make it ideal for emergency response exercises. In the coming months and years, this kind of structured, experience-based exercise will be critical even more critical for communities that are struggling with emergency readiness. Web enabled exercises may allow users at different levels of response (local, state, federal) to participate in distributed exercises without traveling to a common exercise venue.

The cost and complexity of distributed simulation as it exists today is a barrier to widespread distributed simulation development and use. In particular, the emergency response community is in need of tools that will improve readiness for disasters at all levels, but at the current rate of technology transfer it will be many years before distributed simulation can be found in community firehouses and police stations. It is important that the needs of these user communities be considered when developing new technologies for distributed simulation.

8. References

Clarke, T.L. (Ed) (1995) "Distributed Interactive Simulation Systems for the Simulation and Training in the Aerospace Environment" *SPIE*, Bellingham, WA. pp.327

McDowell, P. an Darken, R. (2004) "Using Open Source Game Engines to Build Compelling Training Simulations," *Proceedings of The Interservice/Industry Training, Simulation and Education Conference (I/ITSEC),* Orlando, Florida.

Kincaid, J., Donovan, J., Pettitt, B. (2003) "Simulation techniques for training emergency response." *International Journal of Emergency Management,* vol. 1, no. 3.

Smith, R. (2003). "The Application of Existing Simulation Systems to Emerging Homeland Security Needs." *Proceedings of the 2003 European Simulation Interoperability Workshop,* Stockholm, Sweden.