SIMNET: The Advent of Simulator Networking

DUNCAN C. MILLER, MEMBER, IEEE, AND JACK A. THORPE

Invited Paper

SIMNET was the first successful implementation of large-scale, real-time, man-in-the-loop simulator networking for team training and mission rehearsal in military operations. This paper provides some historical background on how SIMNET was developed within the US Department of Defense, and outlines the key philosophical and architectural principles on which it was based.

The SIMNET battlefield simulation was sponsored by ARPA (then called DARPA), in partnership with the US Army, and was developed and implemented between 1983 and 1990. The emphasis of SIMNET from the outset was on enhancing tactical team performance by providing commanders and troops an opportunity to practice their skills in a dynamic, free-play environment, in which battle outcomes depend on team coordination and individual initiative, rather than on scripted scenarios controlled by an instructor. This program demonstrated the feasibility of linking together hundreds or thousands of simulators (representing tanks, infantry fighting vehicles, helicopters, fixed-wing aircraft, etc.) to create a consistent, virtual world in which all participants experience a coherent, logical sequence of events. In this world, the causal connections among these tactical events, from the individual crew station to the battalion command post, are clear and easily inspectable.

The SIMNET architecture and protocols have evolved into the Distributed Interactive Simulation (DIS) Standard Protocols (IEEE 1278-1993 and its successors), and have provided the foundation for a new generation of battlefield simulations for training, mission rehearsal, tactics development, evaluation of hypothetical new battlefield systems, and concept testing and evaluation.

The authors are two of the principal architects of SIMNET: Thorpe was the original DARPA SIMNET program manager, and Miller formed and led the group at Bolt Beranek and Newman, Inc. (BBN) that designed and implemented the SIMNET protocols and software.

I. BACKGROUND

Speculation about the feasibility of distributed, interactive simulation began in the mid-1970's as the advantages of simulation were becoming better understood. During this period, the development and application of simulation technology was dominated by engineering research and development centers, rather than training centers. The driving force behind simulation improvements was a need to accurately model the platforms being simulated and the environments in which they operated. Advanced simulators were successfully employed in the design and engineering of many new systems.

As simulators improved, the training community began to adopt them. Many of the same engineering professionals who had built simulators for R&D applications adapted their designs for training applications. The most well known applications were in civilian airline and military pilot training.

Most uses of simulators in flight training were for substitution tasks: Training which had been accomplished in the air, but now was relegated to simulators for reasons of cost, danger, security, or nonavailability of time or space. Simulation was seen as a substitution for flight time, and simulators were paid for by reductions in flying hours.

This training focused principally on acquiring the skills needed to operate one's own individual vehicle. For example, airline operation was concerned with the flight profile of the solitary airliner, sometimes but not always within the supporting context of the air traffic control system. Military training focused on undergraduate pilot training or training for the pilot's transitioning into new aircraft. Both used simulators for recurrent training of emergency procedures. However, training involving multiple aircraft in a tactical context was conducted in actual aircraft on ranges rather than in simulators because of the complexity of the task and because means for connecting multiple simulators were as yet undeveloped, except for a few cases of one-on-one air combat (two simulators connected at the same facility). In other domains, such as training maritime ship's captains, the situation was similar. The emphasis was on acquiring the essential skill set needed for the safe operation of one's own platform.

Simulator training often consisted of a sequence of scenarios, monitored and mediated by an instructor at a control console. When other aircraft or ground vehicles appeared in a scenario, they did so as computer-controlled targets, or perhaps in the role of simulated friendly platforms following a predefined script. Because of limitations in the simulator's processing capabilities and, especially, its limited visual display systems, there were rarely more than

Manuscript received November 3, 1994; revised March 3, 1995.

D. C. Miller is with the MIT Lincoln Laboratory, Lexington, MA 02173 USA.

J. A. Thorpe is with the Science Applications International Corporation (SAIC), San Diego, CA 92121 USA.

IEEE Log Number 9411684.

a few such external entities involved in a scenario. To the extent that these entities responded to the actions of the crew being trained, it was either a result of some simple preprogrammed logic or because the instructor intervened to invoke an alternative behavior.

In the late 1970's the value of simulation for nonsubstitution training began to be considered by the military. It was argued that the greatest contribution from simulation would come in areas where expert performance was essential to successful combat operations, but where such expertise was unobtainable using conventional training approaches, even if constraints of cost, danger, security, space, and time were not factors.

In some cases, this expert performance represented the set of combat tasks that would be expected to be performed on the first day of any hostile combat operation, but which were never practiced in peacetime in real vehicles. Flying very low to the ground in dogfights or avoiding a missile closing in at high speed are examples.

In other cases, the needed expertise was the coordinated performance of a large number of combat forces working together to achieve a integrated battle outcome, especially where the battle involved a mix of weapon systems, organizations, and perhaps nationalities, which had to function together on the battlefield against a hostile, sentient opponent. These skills are referred to as collective skills, as distinguished from the mastery of the operation of one's individual platform. Conducting large, coordinated operations in the uncertainty of a battlefield is probably the most challenging activity humans are expected to perform. Doing it poorly can mean loss of life and mission failure.

The debate was based on an argument characterized in Fig. 1, which shows the relationship between expertise in collective skills, practice required to achieve such expertise, attrition in combat without that expertise, and the ability of typical military training programs to build that expertise. The principal tenets of this argument are as follows.

1) Developing expertise in collective skills, i.e., developing the ability for teams to operate well together, is extraordinarily difficult given the complexities of battle. It demands great amounts of practice, as does the acquisition of any complex skill.
2) The opportunities to acquire this practice using combat platforms on ranges during peacetime is severely limited, and in many cases nonexistent for some highly dangerous operations, or completely impractical for units that are widely separated and that rarely practice together.
3) Deficiencies in these collective, team skills result in high casualty rates during combat, much higher than casualty rates attributable to deficiencies in individual skills.
4) Simulation is best targeted at offsetting the deficiencies in collective training by augmenting the opportunities to practice collective skills, i.e., to provide opportunities for attaining mastery of high attrition collective skills (those that must be practiced frequently to maintain performance levels) essential for success in combat.

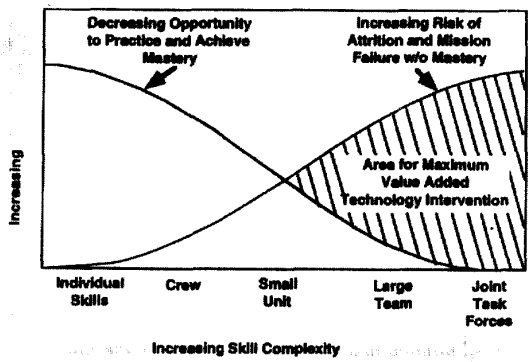


Fig. 1. Collective skill acquisition versus scope of activities.

Providing the simulation infrastructure necessary to support such a capability, however, was a daunting challenge. A white paper written by Thorpe at the Air Force Office of Scientific Research in 1978 described the functional nature of a networked simulation technology to meet this need, but could not specify the engineering approach to accomplish the goal. Simulators during this period were very expensive, special purpose, and limited in functionality. Further, the enabling information technologies that were being developed in other areas, such as packet switching on the ARPAnet, the precursor to the Internet, had yet to influence simulator design.

A few years later this situation began to change. Microprocessor technology began to produce relatively inexpensive computers that could execute much of the code needed for high fidelity algorithms. Affordable local and wide area networks became available to connect these computers. Real time computer image generators (graphics engines) began to show the favorable cost/performance trend that is still under way today. Finally, a design approach for simulators based upon an analysis of the information (cues) the simulator needed to deliver to the users so that a task could be performed properly allowed a simulator design to be tailored to the task and audience, often resulting in the elimination of unnecessary components from the simulation. For the first time, it became feasible to consider developing very large networks of simulators that could be connected within and between sites for operation in real time, and that could be manufactured at a low enough cost without sacrificing fidelity that ample numbers could be connected to create an environment of sufficient complexity to meet true collective, joint, multinational coalition training and readiness needs.

The challenge to pioneer this development was accepted in the spring of 1983 by DARPA (now ARPA), and was called the SIMNET (SIMulator NETworking) Project. The project quickly attracted the attention of the US Army, which provided substantial cofunding and resources (troops, sites, and experts).

II. SIMNET

Initially, SIMNET was aimed at providing the enriched practice environment for complex collective skills training. The two key challenges were 1) how to fabricate high quality/low cost simulators and 2) how to network them together to create a consistent, virtual battlefield. The problems experienced in developing the ARPAnet suggested that a testbed approach for SIMNET would be the only way to successfully develop and debug a simulation network. The project was scoped to construct a testbed consisting of at least four sites with 50–100 vehicle simulators at each site. This scale was judged suitable for testing the networking architecture and techniques.

Because it was thought that networking several hundred flight simulators would be too demanding of networking capacity, given the speed and maneuverability of aircraft (a speculation that turned out to be fallacious), the project began by developing and networking slower-moving ground vehicles (tanks and armored personnel carriers) with the plan to scale up to helicopters and then to fighter aircraft if satisfactory results were achieved.

The SIMNET simulators were the first of a class designed around a principle articulated by Dr. Bob Jacobs (then of Perceptronics, Inc.) called selective fidelity. This principle began with a functional specification of what the simulation was expected to do, what information was needed by the user, and what he would do with that information. The essential components of the simulator could be identified from this analysis, along with the minimum level of fidelity required, and the less essential and nonessential components of the simulation could also be identified. Simulators would be designed accordingly, with some very high fidelity components, some moderate fidelity components, some abstracted components, and some components that were not represented at all.

As the program commenced, the two research teams under contract to DARPA, Bolt Beranek and Newman, Inc. (BBN) of Cambridge, MA, and Perceptronics, Inc., of Woodland Hills, CA, began to interact with subject matter experts at the US Army Armor Center at Ft. Knox, KY. As the potential of the technology became better understood at Ft. Knox, the focus of the application was extended from training to development and evaluation of innovative ideas for next-generation combat systems.

In essence, the emerging idea was that a large scale, interactive, networked simulation created a synthetic environment (or virtual battlefield) that could be entered by any authorized combatant from anywhere on the network using his simulator as a porting device. Once there, he or she would interact with others who had similarly ported themselves onto that battlefield in a free play, nonscripted setting, constrained only by the operational rules and disciplines to which they subscribed (the chain of command and its rules of engagement).

Such synthetic environments or artificial worlds were independent of the application to which visitors subjected them. One could use the virtual battlefield for training, mission rehearsal, concept development, doctrine and tactics development, testing, after-mission review, or whatever purpose the environment might be good for. If several individuals or groups wished to use the network at the same time for their own purposes without interference from others, it was possible to create and sustain several separate, independent worlds simultaneously.

III. SIMNET Milestones

After the basic design philosophy and system architecture of SIMNET were developed, as described in the following section, the first conceptual demonstration was conducted in December 1984. The first demonstration involving real-time, out-the-window graphics was conducted in November 1985. The first platoon-level system, incorporating custom image generation capabilities designed to optimize performance in an environment with large numbers of moving objects, was installed at Ft. Knox, KY, in April 1986. The first helicopter simulators were installed at Ft. Rucker, AL, in late 1987.

At the conclusion of the program, a network of approximately 250 simulators, installed at nine operational training sites (including four in Europe) and two developmental sites, were transitioned to the US Army. Two mobile platoon sets, installed in containers mounted on flatbed trailers, were delivered to the Army National Guard.

The communications protocols by which the simulators transmit essential information to one another were subsequently incorporated in the Distributed Interactive Simulation (DIS) Standard Protocols (IEEE 1278-1993), approved on March 17, 1993 [1]. The next generation of this standard is currently under development. Participants from government, industry, and academia convene for semi-annual conferences and workshops in September and March of each year in Orlando, FL, to discuss and create these protocols and the necessary supporting technology and practices.

IV. SIMNET Protocols

The success of the SIMNET approach [2], and the fact that all of its essential elements were adopted into the DIS standards after intense scrutiny within a highly competitive industry, was the result of several key design principles and architectural decisions that were made early in the program and that were then progressively refined. These principles are summarized in the following paragraphs.

A. Object/Event Architecture

The first key decision was to model the world as a collection of objects, which interact with each other through a series of events. The basic terrain and cultural objects (buildings, bridges, etc.) are assumed to be known to every other object. This paradigm permits the possibility of an event that changes one or more of these objects, e.g., destroying a bridge, or digging a pit at a certain location, though these capabilities were not implemented under the SIMNET program.

B. Autonomous Simulation Nodes

All events are broadcast on the simulation network, and are available to all objects that may be interested in them. The simulation nodes that initiate an event do not need to keep track of what other nodes may be affected by that event; these calculations are the responsibility of the receiving nodes.

In SIMNET (and DIS), there is no central control process that schedules events or resolves conflicts among contradictory versions. Instead, each simulation node is completely autonomous; the communications algorithms even permit a simulation node to join or leave an exercise in progress without disrupting the interactions among the other nodes.

Each node is responsible for maintaining the state of at least one object in the simulated world, and for communicating to other nodes any events caused by its object(s). Each node is also responsible for receiving event reports from other nodes, and calculating the effects of this event on the object(s) they are simulating. If the effects cause other events to occur, then the node is responsible for notifying others of these events.

1) Transmission of "Ground Truth" Information: Each node transmits absolute truth about the current state of the object(s) it represents and any events they have caused. It is the responsibility of the objects receiving an event message to determine whether they are able to perceive the event and whether (and how) they are affected by it. When information needs to be degraded before presentation to a human crew or an automated crew (e.g., on a radar display), it is the responsibility of the receiving objects to calculate and introduce this degradation.

2) Transmission of State Change Information: To minimize communications processing, nodes transmit state update information only when the object(s) they represent change their behavior. A change of behavior constitutes an event that may be of significance to other objects. This approach minimizes repetitive transmission of redundant information, which substantially reduces the processing load on other nodes.

C. "Dead Reckoning" Algorithms

Between state update messages, receiving nodes extrapolate the last reported states of remote objects that are of interest to their local object(s), and use this information in generating displays for human crews or detection probabilities for automated crews. The sending nodes are responsible for generating a new state update message before discrepancies in the remote extrapolations become unacceptably large.

In effect, this algorithm depends on a "contract" among the nodes. Each node guarantees that it will transmit a state update event within 1/15 s (or some other agreed-upon interval) of the time that the true position and orientation of any object(s) it represents diverges from the calculated values (based on an extrapolation of the last reported state update information) by more than an agreed-upon threshold. Obviously, this algorithm requires that each node maintain a dead reckoning model that corresponds exactly to the model(s) being used by the remote nodes.

Since each state update includes corrected position, as well as velocity and heading information, the dead reckoning algorithm is essentially self-healing. A node that fails to receive a state update message will, at worst, continue to extrapolate the previous state of a remote object for a few additional seconds. If the remote object continues to change its behavior, it will generate a burst of updates, and a new message is highly likely to arrive within a fraction of a second. This new message will correct any error that has accumulated and will initialize a new extrapolation.

In SIMNET, simple position, orientation, and velocity information was used for all vehicles. Manned vehicle simulators recalculated state information 15 times/s. Actual update transmission frequencies averaged one per second for ground vehicles and three per second for air vehicles, although individual vehicles often transmitted 15 updates per second during periods of intense activity. Tradeoff studies showed that the use of second derivative state variables reduced transmissions for air vehicles to about one per second as well [3].

V. Types of SIMNET Simulations

Three principal types of simulations were incorporated in the SIMNET architecture. In addition, other nodes were used to support after-action review and data analysis.

A. Manned Vehicles

Manned vehicle simulations are intended to provide realistic control/display interactions for each crew member. For example, an M1 tank simulator provides controls and out-the-window displays for the tank commander, gunner, loader, and driver. An AH-64 helicopter simulator provides controls and displays for the pilot and the copilot/gunner. Each manned vehicle crew must navigate, avoid obstacles, maintain formation, detect and identify targets, employ weapons, and communicate via radio and other devices, just as they would on the battlefield.

B. Command Post Simulations

Command post simulations focus on decision-making and resource allocation, rather than on moving and shooting. They do not require out-the-window displays or detailed control/display interactions.

Typically, command post simulations involve radio/telephone communication, a map display, and a workstation that supports the types of decisions and inputs that are normally made by each functional position. A Fire Support Officer, for example, uses a display that shows the location and status of his artillery batteries, his remaining ammunition supply, and currently scheduled fire missions. These missions can be interrupted or modified, and new missions can be added, as circumstances require. A logistics support officer has a limited number of fuel and ammunition carriers, which he can dispatch to selected

locations to rendezvous with and resupply units that are running low on expendables.

C. Semi-Automated Forces (SAF)

Semi-automated simulations are designed to realistically mimic the externally visible behavior of opposing or supporting forces without requiring large numbers of manned simulators and personnel to operate them. A human commander exercises supervisory control over units that may include dozens of vehicles. These multivehicle simulations broadcast the same state update messages as manned simulators, so that their behavior is indistinguishable to the other simulators and (usually) to their crews. The semi-automated simulations of vehicles and small units are intelligent enough to provide basic route planning, obstacle avoidance, formation keeping, line-of-sight detection of nearby vehicles, target engagement, and so forth.

The human commander provides the goals and objectives for his subordinate units via input forms modeled after standard field orders and graphic overlays. He continually monitors the forces under his control, and can intervene to redirect their activities at any point. He may temporarily assume direct command of a lower-level subordinate unit (e.g., a tank platoon) in order to exercise finer control over their actions.

The SAF commander's workstation consists of 1) a task organization display, showing how the units currently under his command are organizationally related to each other, 2) an operations display for composing orders and requests for information, 3) a message log display, which displays recent radio reports from the units under his command, and 4) a military map display on which the latest position reports, enemy contact reports, etc., are automatically posted.

The SAF commander also maintains radio contact with command post staff and commanders of fully manned units.

D. Other Simulation Nodes

Other key elements in the simulation network are designed to observe exercises in progress and/or to collect data for later analysis and replay. The most important of these elements are the data collection and analysis system and the Observation Vehicle (or "Flying Carpet").

The data collection and analysis system captures, time stamps, and records every protocol data unit (PDU) transmitted by each node. Using the time stamp information, any portion of any exercise can be replayed onto the network at a later time, and any simulator can be used to drive or fly freely throughout the battlefield, seeing every event that one would have seen at the time the exercise was conducted. In effect, this permits a "time travel" capability—the ability to view perfectly reconstructed events, which are not affected by the presence of the time-traveling simulator. The data, once recorded, can also be analyzed by a suite of statistical analysis tools to prepare reports regarding what took place.

The "Flying Carpet" is a simulator that provides both a situation (map) display and an out-the-window view of the battlefield. It is invisible to other simulators, and so can be used to observe an exercise in progress without affecting it, as well as for "time travel." The Flying Carpet can be attached to, and "towed" by any selected vehicle, so that the behavior of that vehicle can be monitored in detail.

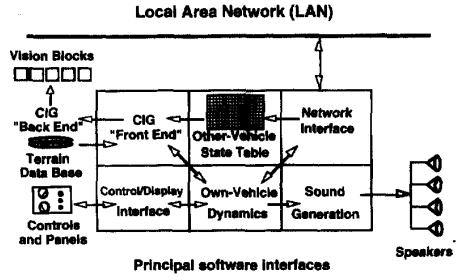


Fig. 2. Software modules in a typical simulator.

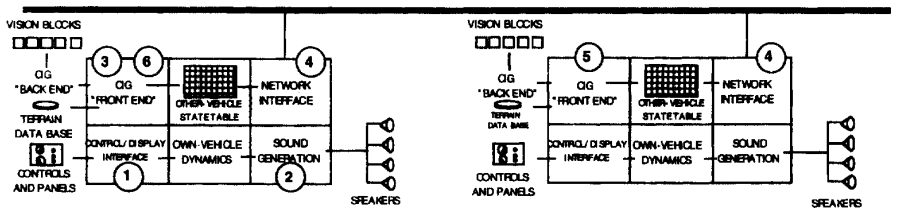
VI. An Example of SIMNET Protocol Interactions

To appreciate the kinds of cooperative interactions involved in DIS simulations, consider a simulator that includes the following software processes (See Fig. 2).

1) A network interface, that sends and receives DIS PDU's on a local area network (LAN).
2) An "other vehicle state table" that records entity state information from other simulation nodes and carries out dead reckoning extrapolations based on their last reported states.
3) A computer image generator (CIG) front end, which combines the nonchanging information from the local copy of the terrain data base and the dynamic information from the other-vehicle state table, calculates the potentially visible terrain and dynamic object polygons, and transfers them to special-purpose CIG back-end hardware. This specialized hardware filters, clips, rotates, and scales visible polygons, applies calculated texture patterns, and sends them to the proper vision block displays for viewing by the crew members.
4) An own-vehicle dynamics model that computes (in whatever detail is necessary) the state of the entity being simulated.
5) A control/display interface process that reads control positions and drives meters, gauges, and similar displays.
6) A sound generation process that drives a set of speakers within the simulator.

Now imagine two such simulators, and let's step through the sequence of events that occurs when simulator A fires at, and hits, simulator B (See Figs. 3 and 4).

1) First, the control/display interface detects that the gunner has pulled the trigger.
2) The sound generation software is notified that a local main-gun sound effect is required.



Sequence of events when Simulator A fires at Simulator B

1. **Trigger pull detected**
2. **Local sound effect produced**
3. **Local muzzle flash produced**
4. **Entity State PDU sent; received at other nodes**
5. **Muzzle flash displayed by other simulators**
6. **Ballistic flyout calculations occur; tracer image displayed**

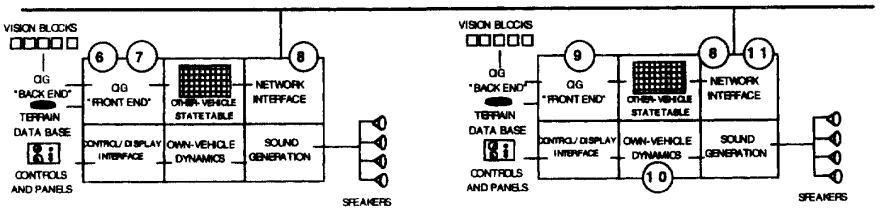
Fig. 3. Typical sequence of events when Simulator A fires at Simulator B.

3) The image generator front end is notified that a local muzzle flash effect is required, which will obscure the gunner's and commander's vision blocks for a second or so.
4) The network interface process is notified that an event has just occurred that has changed the vehicle's appearance, and a new Entity State PDU is transmitted onto the network. This PDU is received by all other simulation nodes, and the updated state information is transferred into the other-vehicle state tables.
5) During the next CIG frame recomputation, the updated state information for simulator A is incorporated, and (unless some intervening object blocks his view) any crew member looking in the direction of simulator A will see a tank with a large fireball at the end of its gun tube.

   Note that at this point no calculation has yet begun as to where the round that was fired is going. We will assume for the purposes of this explanation that the round follows a predictable ballistic trajectory. For a guided munition, the story is more complicated and beyond the scope of this paper.

   Well in advance of the simulation, flyout trajectories have been computed for each type of ammunition that can be fired by a given simulator.

   Each flyout trajectory is then divided into short chord segments that correspond to the distance traveled by the projectile within one frame interval.
6) When the round is fired, the gun tube vector is used to initiate a flyout calculation. As the CIG computes each new out-the-window view, it retrieves the next sequential chord segment from the flyout vector. It computes whether this chord segment intersected any polygon in its static or dynamic object list. If not, it displays a tracer image for the crew of the firing simulator. If so, it returns the x, y, z coordinates at which the intersection occurred and the object with which the polygon is associated. Note that this approach ensures that a vehicle that inadvertently drives or flies into the path of a projectile will be hit.
7) Once an impact has been determined, the CIG displays an appropriate impact effect to the crew of the firing simulator. The effect displayed for a kinetic energy round striking the ground is different from the effect of a high-explosive round striking armor.
8) The network interface software transmits a Detonation PDU, which is received by all other simulation nodes. This information is inserted into the other-vehicle state table for incorporation in the next out-the-window image.
9) The detonation is displayed. Crews of any vehicles in the vicinity that have an unobstructed line-of-sight to the detonation point will see the round impact effect. For a simulator that does not represent an entity struck by the round, this is all that is required.
10) If, however, the simulator finds that the impacted polygon is part of an entity it represents, it is responsible for determining what damage, if any, it suffered as a result of the impact. Just as the simulator representing the firing entity is responsible for determining the point of impact, the simulator representing the struck entity is responsible for determining the effect of the impact. The simulator typically uses probabilistic damage tables based on ammunition type, where it was struck, the angle of incidence, and the firing range. Damage results can range from none at all to a catastrophic kill.



Sequence of events when Simulator A fires at Simulator B

6 Ballistic flyout calculations occur; tracer image displayed
7 Impact signature displayed locally
8 Detonation PDU transmitted
9 Detonation effect displayed at other simulators
10 Damage effects calculated by target vehicle
11 Visible damage effects (if any) transmitted in Entity State PDUs

Fig. 4. Typical sequence of events (continued).

11) In the case of a catastrophic kill accompanied by fire, the simulator representing the struck entity will begin broadcasting Entity State PDU's that reflect a burning vehicle at its current location.

VII. SIMNET Simulation of Radio/Telephone Communications

One other instructive example is the SINCGARS radio simulation developed for the Army Research Institute (ARI) Combat Vehicle Command and Control system simulation effort, described briefly below. This simulation involved the propagation of digital map data, as well as voice signals, over a standard radio channel.

First the voice and data signals were digitized at the simulated transmitter. Header information was added to the data packets to represent the transmitter location, frequency, power, and antenna orientation.

Using principles similar to those for the effects of projectiles, the receiver simulators were responsible for determining which signals were received by the crew. Each receiving simulator used its copy of the terrain data to compute the received signal strength, based on the known location of the transmitter and receiver and diffraction effects caused by intervening objects. A receiver characteristics model was used to determine which signals were captured; only the "winning" signal was decoded and mixed with noise at the calculated signal-to-noise ratio. This signal could be a jammer, as a result of either intentional or unintentional jamming.

The same signal source information could be used for navigational purposes (by friendly forces) or by radiation-seeking missiles (to disrupt enemy communications).

VIII. SIMNET Applications

Since the initial emphasis of SIMNET was on tactical team training, it is not surprising that the large majority of SIMNET applications have been training exercises. These are conducted at battalion level (80 manned combat vehicles, plus 200–300 SAF entities) at two sites: Ft. Knox, KY, and Grafenwoehr, Germany. The other sites run company-level exercises (up to 20 manned combat vehicles, plus 50–100 SAF entities). These training exercises are typically run in stand-alone mode, without Wide Area Network links among the sites.

As the SIMNET simulators and protocols matured, ARPA began a series of cooperative developments and exercises with other DoD organizations aimed at the evaluation of hypothetical weapons systems, tactics development, and rehearsal for field test and evaluation exercises. The following paragraphs summarize a few of the most significant efforts.

A. Forward Area Air Defense System (FAADS)

In 1988 and 1989, the Army Air Defense Artillery School at Ft. Bliss, TX, sponsored two exercises aimed at rehearsing field tests of the Forward Area Air Defense System (FAADS) then under development. In a rapid development effort, ARPA contractors electronically linked a simulated FAADS vehicle turret located at one simulation node to a modified M2 chassis at another node to create a single vehicle that closely approximated the FAADS vehicles. The FAADS fire control system required the rapid development of a digital radar display simulation, which necessitated a third network interface per simulated vehicle. Some new protocol data units (PDU's) were required to support target handoff functionality among the FAADS platoon

vehicles. New PDU's were also required to transmit radar antenna orientations, in order to support simulated Radar Warning Receivers in the participating aircraft simulators. The FAADS exercises uncovered several problems that would otherwise have not been discovered until expensive live-range tests were under way. They also yielded several useful insights that resulted in modifications of planned tactical doctrine for the new system.

B. Combat Vehicle Command and Control (CVCC)

This study was sponsored by the Army Research Institute (ARI) at Ft. Knox beginning in 1988. It involved the simulation and testing of a suite of advanced devices, some of which were subsequently incorporated in the M1A2 main battle tank upgrade program. This study required the development of simulated thermal imaging devices to represent the commander's independent thermal viewer (CITV). It also required the development of touch panel displays for scrollable digital maps inside the tank's turret, with text and graphic electronic overlay images that could be transmitted to other vehicles and to battalion headquarters by means of digital message transmission over simulated SINCGARS radios. It required the development of the radio simulations described previously. "Touch panels" were used for direct entry and dispatch of tactical spot report information. A position/navigation (POS/NAV) system was simulated that allowed tank commanders to input waypoint information, as aircraft pilots do, to direct his driver along a chosen route. Extensive studies were conducted by ARI and its supporting contractors, which led to many lessons that were incorporated into the systems installed in the M1A2.

C. Nonline-of-Sight (NLOS) Missile

In 1989 and 1990, the US Army Missile Command sponsored a rapid-turnaround simulation of a proposed NLOS Missile system. Because of the short time frame involved, this development was based on the SIMNET "flying carpet" vehicle, which was modified to support multiple missiles in flight, with the ability to switch a camera eyepoint from missile to missile. A variety of target tracking modes were simulated and compared.

D. Counter Target Acquisition System (CTAS)

From 1990 to 1991, a DARPA-sponsored consortium including the Los Alamos National Laboratory assessed the effects of laser weapons on the battlefield, which could be used to disrupt sensors and disorient pilots. This study required the development of head/eye tracking apparatus to discern precisely where a pilot is looking when painted by a laser beam. Government-furnished models of laser effects were used for simulating canopy glare; sensor disruption, including vidicon tube blooming and pitting; and eye damage effects. New protocols were developed to support these effects and to facilitate data collection. This study is a prime example of one that could not reasonably have been carried out in a field test because of the dangers involved.

E. Line-of-Sight Anti-Tank Missile (LOSAT)

In 1990, the US Army Missile Command sponsored another study of a weapons system under development, which required further protocol and simulations extensions to support multiple missile firing and tracking. The principal challenge of this study was the implementation of automatic target recognition algorithms and target queuing algorithms that are based on estimated threat of a particular target.

1) Navy Battle Force In-Port Trainer: In April 1990, an existing Navy surface/air simulation called the Battle Force In-Port Trainer was interfaced to SIMNET by ARPA and Navy contractors. An exercise was conducted in which aircraft simulators at Ft. Rucker, AL, appeared on the radar screens of real ships docked in Dam Neck, VA, and Charleston, SC. Simulated naval gunfire from these ships affected ground forces in manned simulators at Ft. Knox, KY. This exercise marked the first time two virtual simulations, developed independently for different purposes, had been made to interact.

F. Battle Reenactment

After the Gulf War, SIMNET SAF were used to develop an unprecedented shot-by-shot recreation of a key armor/cavalry battle against the Iraqi Republican Guard. Called "73 Easting" from the line of longitude at which it occurred, this battle involved a cavalry squadron effectively destroying a dug-in armored division, several times its size.

In recreating this battle, data were collected from troop interviews, intelligence sources, command radio net recordings, and direct measurements taken on the ground after the battle using global positioning system (GPS) receivers. In several cases, the sources of specific shots were determined by such techniques as tracing the TOW missile fiber optic strands that stretched across the battlefield.

SAF tasking orders were used to generate "best-fit" time histories, which were then observed and critiqued by the troops who had participated in the battle. In a series of iterations, inconsistencies in the reports and the reenactment were uncovered, debated, and resolved. As this process converged, the results were endorsed by the troops as a highly accurate recreation of what happened to each vehicle and unit. Military historians have hailed this recreation as a new benchmark in what is possible for the analysis and review of an engagement.

IX. From SIMNET to Distributed Interactive Simulation (DIS)

With support from the US Army Simulation, Training, and Instrumentation Command (STRICOM), the Defense Modelling and Simulation Office, and (initially) ARPA, a series of semi-annual government/industry conferences and workshops were initiated in September 1989, aimed at building on the SIMNET results to generate a consensus standard for distributed simulation.

The first few conferences necessarily focused on educating the simulation community regarding the SIMNET paradigms. By early 1991, several hundred people were

participating, and revisions and extensions of the SIMNET protocols were actively being debated. By late 1992, agreement was reached on an initial set of DIS standards to be balloted as an IEEE industry standard, and in March 1993, the first standards were formally approved [1].

A. Coordinate System Changes

While the basic structure of the SIMNET PDU's, as well as the key architectural principles, were adopted in the DIS standards, there are some significant differences between the contents of SIMNET and DIS PDU's. The most important differences occur in geographic coordinate systems and angular measurements, but there are also numerous differences in field layouts, numbers of digits, padding within PDU's, and in the enumeration of entity types.

In SIMNET, terrain is represented using Cartesian coordinates based on a local "flat earth" model. The origin of the coordinate system is a selected point at the southwest corner of a (usually) rectangular database. For the larger exercises contemplated for DIS, it was clear that curvature of the earth must be taken into account. After substantial discussion, a Cartesian system with an origin at the center of the earth was selected as the standard for PDU data transmission.

In SIMNET, rotation matrices were used for angular representations and transforms, as is common in the graphics imaging community. To many workshop participants, vehicle body-centered Euler angles were a more familiar representation. The latter view prevailed and was incorporated into the standards.

X. Network Performance Requirements

To support the time-critical interactions required for SIMNET and DIS, the network connecting the various simulation nodes must provide a certain level of assured services. In particular, it must exhibit the following characteristics.

A. Low Transport Latency

The SIMNET rule of thumb was that total latencies should not exceed typical human reaction times (250 ms) or anomalies in causality could become apparent to the participants, compromising the integrity of the virtual simulation. After much discussion, this principle was adopted for the DIS standards. The DIS Guidance Documents [4] specify a total end-to-end latency of less than 300 ms (from the trigger pull to the remote display of the muzzle flash, in the example given previously) for "loosely coupled" interactions, and 100 ms total latency for "tightly coupled" interactions (e.g., formation flying), in which anomalies would be more immediately apparent.

1) Low Latency Variance: Low latency variance is important to minimize jitter in the displayed entity positions, although within certain limits, the required time stamps in Entity State PDU's can be used to compensate for latency variance by inserting corrective changes in dead-reckoning extrapolations.

B. Reasonably Reliable Delivery

The dead reckoning algorithm was designed to be robust with respect to missing datagrams, within certain limits. If an Entity State PDU is missed, the receiving simulator will continue to dead reckon the entity along its prior trajectory until a new PDU is received. If the entity is maneuvering significantly, the next update will arrive within a fraction of a second, the entity's position will be corrected, and the new derivatives will be used to initiate a new extrapolation. A 1% or 2% datagram loss will present no problem, as long as the missing datagrams are randomly distributed and do not occur in correlated sequences.

In SIMNET, standard Ethernet networks were used for all local area network (LAN) connections. Except for the very largest exercises, Ethernet bridges were used via 56 kbps dial-up links to connect two or more LAN's to form a single, logical Ethernet LAN. Except for a few demonstrations, encryption was not used or required. In some more recent DIS exercises, NSA-approved encryption devices have been required. This currently represents a major network bottleneck, with the encryption devices introducing both significant throughput limitations and latencies.

XI. Future Challenges

A. Supporting Very Large Exercises

The largest exercise during the SIMNET program was conducted in March 1990 at five sites. At the peak of the exercise, approximately 850 entities were active, most of which were SAF. Since SIMNET PDU's contain approximately 1000 b and active entities average 1 PDU/s, the traffic volume at the peak was approximately 850 kb/s. DIS PDU's are larger than SIMNET PDU's (typically 1700 b), so the same exercise in DIS would generate roughly 1.5 Mb/s.

If such an exercise were scaled up to involve 100 000 entities, as is currently being proposed as a goal, each simulation node would have to process over 175 Mb/s, extracting from this torrent the relatively small percentage of events that are of potential interest to the entities being simulated at that node. Such a flow would be beyond the capacity of most low-cost simulators, and in any case would represent a very inefficient use of computing and communication capacity. Fortunately, some promising techniques are available to improve this situation.

B. Terrain Compatibility

Although the DIS standards prescribe the protocol and PDU information content required for heterogeneous simulations to interact with each other over a compliant network, this does not guarantee that their interactions will appear realistic or logically consistent. Another key element needed to achieve compatible interactions is the digitized terrain/object database that describes the virtual battlefield. This database takes the form of polygonalized terrain that describes the geometry of the ground and its composition (paved surfaces, hard-packed earth, sand, shallow water,

etc.), plus descriptions of natural and manmade objects on the terrain (tree lines, forested areas, power lines, roads, buildings, etc.)

The problem is that unless two heterogeneous simulators are using geometrically identical representations of the terrain and related objects, tactically significant discrepancies can easily occur. Consider, for example, a situation in which one vehicle has selected a position behind a tree line on a hill in order to be able to observe the battlefield while concealed from enemy view. If an opposing vehicle's simulator does not have an identical representation of the position of the tree line or the height of the hill, it may have an unobstructed line of sight to the position of the vehicle that believes it is hidden in the trees. Worse yet, when a vehicle reports its x, y, z coordinates as part of an Entity State PDU, another vehicle may believe that the reported coordinates are several feet above or below its representation of the terrain at that location. In this case, a tank may appear to be flying or burrowing as it traverses the terrain!

The obvious solution to this problem is to make sure that are participants are using geometrically identical databases. The problem with this solution arises from the fact that these databases are tightly coupled to the computer image generation systems used by the simulators, and that different representational algorithms are used for different vendor's systems. These algorithms are often deeply rooted in the image generator's design, and are therefore difficult and expensive to change.

For these reasons and for other technical reasons, it has proved much more difficult to arrive at a consensus standard for terrain/object representation than it has for the communication protocols. This problem is still being attacked.

C. Command and Control Representation

A major challenge is how to realistically represent command, control, and communication on the virtual battlefield. The flow of information on the battlefield is at least as important to simulating the interactions of opposing forces as the representation of the physical phenomena. All the discussion in previous sections of this paper has addressed only these physical phenomena. There has been no attempt to represent the military strength of a unit, or the battlefield intelligence information available to its commander, or his tactical plans, or the order he just received or the report he just transmitted. This nonphysical information, which is broadly referred to as Command, Control, Communication, and Intelligence (C3I) information, is the next realm being attacked within the DIS community, and will be essential to the continued improvement of the simulation capabilities developed so far.

XII. Conclusion

The principal focus of this paper has been on the architectural concepts initially developed under the SIMNET program in the early to mid-1980's. We have included enough illustrative vignettes from SIMNET and subsequent applications to show several of the ways these original concepts have been extended to new applications and to incorporate new phenomena, many of which were not contemplated at the time of the original development. The original SIMNET architecture has withstood repeated extensions without fundamental change, which is generally considered a critical measure of the robustness of an architecture. We believe that it has fared well in this regard, and we suspect that it will continue to do so as new phenomena and new dimensions of battlefield representations, as well as nonmilitary applications, are incorporated into the distributed, virtual world that has been built on the SIMNET foundation.

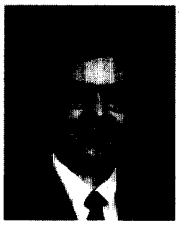
References

[1] IEEE Standard for Inform. Tech.—Protocols for Distributed Simulation Applications: Entity Information and Interaction. IEEE Standard 1278-1993. New York: IEEE Computer Soc., 1993.

[2] A. R. Pope and R. L. Schaffer, "The SIMNET network and protocols," BBN Rep. No. 7627, Cambridge, MA (Revised June 1991).

[3] D. C. Miller, A. R. Pope, and R. M. Waters, "Long-haul networking of simulators," in *Proc. 10th Interservice/Ind. Training Syst. Conf.*, Orlando, FL, Dec. 1989.

[4] S. Seidensticker *et al.*, "The DIS vision," IST-SP-94-01, Orlando, FL, May 1994.



Duncan C. Miller (Member, IEEE) received four degrees from MIT, including the Ph.D. in mechanical engineering.

He currently leads a small group of experts at MIT Lincoln Laboratory specializing in Distributed Interactive Simulation (DIS) architectural issues. This group facilitates cooperation and technical exchange across government programs regarding DIS standards and related issues. From 1963 to 1993, he worked at Bolt Beranek and Newman Inc. in Cambridge, MA. In 1983, he formed and led the group that developed the protocols and software for SIMNET, the innovative ARPA program that led to DIS. For the last five years of his tenure at BBN, he served as Vice President of BBN's Systems and Technologies Division, with responsibility for advanced simulation engineering activities in Cambridge, MA, and Bellevue, WA. His principal areas of study included control theory, human operator performance modeling, human factors, and perceptual psychology.

Dr. Miller is a member of the DIS Standards Coordinating and Steering Committees and chairs the DIS Technical Committee. He also is a member of the ARPA/DMSO Core Team developing recommendations for a common High Level Architecture for DoD Modeling and Simulation. He is a member of the Society of American Magicians, an affiliation that proved unexpectedly useful in developing the principles of distributed simulation.

Jack A. Thorpe received the Ph.D. in industrial psychology from Bowling Green State University, Bowling Green, OH.

He is currently a corporate Vice President at SAIC, San Diego, CA, where he coordinates R&D of advanced distributed simulation projects at the corporate level. From 1981 to 1993, he was an Air Force Officer (he retired as a Colonel) at DARPA, Washington, DC, where he was Program Manager for several advanced simulation technology projects. In 1983 he proposed the SIMNET project, secured Agency and Army funding, and was its program manager. In 1991, he was given the additional responsibility of being one of the seven "thrust leaders" in the DoD Science and Technology strategic reorganization plan.