LONG-HAUL NETWORKING OF SIMULATORS

Duncan C. Miller
Arthur R. Pope
Rolland M. Waters

Simulation and Training Systems Divison
BBN Systems and Technologies Corporation
33 Moulton St., Cambridge, MA 02238
(617) 873–3000

ABSTRACT

This paper describes techniques used to successfully link ground vehicle and aircraft simulators at widely dispersed sites into a common network. This network enables the crews of each of the simulators to see and interact with each other in a realistic battlefield environment. Fully-manned platoon-, company-, and battalion-level units may fight force-on-force engagements in real time over the network. The network design uses a distributed simulation approach aimed at minimizing both communications processing loads and simulator complexity. This approach involves abstracting, thereby reducing, the information that is broadcast among the simulators. It achieves an appropriate three-way balance among computation requirements, accuracy of vehicle state representation, and communication loads.

INTRODUCTION

For the past five years, BBN has been developing methods of linking hundreds of combat vehicle simulators and their supporting elements into a common simulated battlefield. This work has been sponsored by the Defense Advanced Research Projects Agency (DARPA) in partnership with the U.S. Army.[1] The resulting network, called SIMNET, has clearly established the feasibility of large-scale simulator networking, paving the way for the Army's planned Close Combat Tactical Trainer procurement. The network consists of several local area networks at various sites, connected by long-haul links operating over terrestrial lines, satellite transceivers, or other transmission media.

In effect, this real-time simulator network provides a simulated world in which fully–manned platoon–, company–, and battalion–level units can fight force–on–force *engagements against opposing units of similar composition*, without incurring the costs of transporting large numbers of personnel and equipment. The network can be used to support a joint, combined arms environment with the complete range of command and control and combat service support elements essential to actual military operations. All of the elements that can affect the outcome of a battle can be represented, so that victory on the battlefield is likely to go to whichever side is better able to plan, orchestrate, and execute its tactical operations. Whatever the outcome, combat units can benefit from the opportunity to practice their warfighting skills at a small fraction of the cost of an equivalent exercise in the field. In fact, they can practice actions that are too dangerous or too likely to damage expensive equipment when carried out in a field exercise.

The network is currently being used to connect simulators representing M1 Abrams Main Battle Tanks, M2/M3 Bradley Fighting Vehicles, simulators, simple helicopter and fixed–wing aircraft, and generic air defense artillery systems. Additional vehicles are currently being considered for possible inclusion on the simulated battlefield.

Combat Vehicle Simulators

Each combat vehicle simulator connected to the network employs one or more real–time computer image generation (CIG) systems to provide a multiple–window view of the battlefield. Through these windows, the crews see a representation of actual terrain constructed from Defense Mapping Agency data. This terrain is represented with sufficient realism that the crews can navigate through it, recognizing roads, rivers, hills, tree lines, and other distinctive terrain features as they would on actual terrain. In addition, they see combat vehicles, combat support vehicles, and combat service support vehicles—tanks, helicopters, self–propelled howitzers, fuel trucks, etc. The actions of these vehicles reflect the control actions of other vehicle crews in other simulators elsewhere on the network.

Other Simulation Modules

In addition to combat vehicle simulators, several other kinds of modules may also be connected to the network, including

- **Automated combat support and combat service support vehicles, including howitzers, mortars, fuel and**

[1]This work is currently being supported under contract MDA-903-86-C-0309.

Butterfly is a trademark of Bolt Beranek and Newman Inc.

Ethernet is a trademark of the Xerox Corporation.

ammunition supply vehicles, and maintenance vehicles. These vehicles are controlled from consoles attached to the SIMNET Management, Command and Control (MCC) system. The Logistics Console is used by the battalion S–4 Officer to direct fuel and ammunition vehicles to rendezvous points at which the combat vehicles are resupplied. The Fire Support Console is used by the battalion Fire Support Officer to direct artillery and mortar fire. The Close Air Support Console is used by the Air Liaison Officer to direct air strikes in support of battalion operations. The Fire Support Console and the Close Air Support Console are co–located in the battalion Tactical Operations Center (TOC).

- **Semi-automated forces, including armor, mechanized infantry, helicopter, and fixed wing aircraft.** These forces are under the supervisory control of a human commander at all times. He directs their activities, specifying routes and objectives, fire priorities, etc., and then monitors and controls their behavior on a plan view display. A BBN Butterfly™ parallel processing computer is used to run detailed simulations for every vehicle in the unit, and to broadcast and receive state update information in exactly the same formats that fully-crewed vehicles use. The computer runs processes that provide route-following, obstacle avoidance, formation-keeping, target acquisition, and similar functions. If the commander is satisfied with the behavior generated by these processes, he will continue providing only high-level instructions to his unit. He can intervene at any point, however, and redirect the activities of any company, platoon, or even a single vehicle.

- **Data collection and analysis systems, which capture, time-stamp, and log onto a disk file every state update message that is broadcast on the network.** These messages can be replayed onto the network at a later time for the purposes of analysis or after-action review. A simulator can be driven through, or flown over, the battlefield, displaying on its viewports precisely what it would have shown had it been present when the exercise was recorded. It cannot affect actions on the battlefield, of course, since they are now, in effect, pre-recorded. In addition, a powerful set of data extraction and statistical analysis tools can be used to calculate standard measures of unit performance, or to calculate non-standard measures defined for a specific exercise.

When two or more local area simulator networks are connected by long-haul links, as described later in this paper, all simulators can communicate with each other as if they were located at a single geographical site. Multiple exercises can be conducted on the same network; simulators will ignore transmissions from simulators in other exercises. At this point, an exercise is limited to 500 active simulators. Later, when more intelligence is added to the long-haul link gateways, there should be virtually no limit to the size of the exercise that can be conducted.

Local Area Networks

The local area network (LAN) used at all sites is an Ethernet™ operating at ten megabits per second. This choice was made for several reasons: Ethernet interfaces are inexpensive, available from multiple vendors for essentially every make of computer, and support both broadcast and point-to-point message transfer with automatic error detection. Furthermore, Ethernet efficiently handles the 1024-bit datagrams employed for state update messages (which are described later in this paper) at the data rates necessary for simulator communication. In terms of the ISO Basic Reference Model for Open Systems Interconnection, the messages communicated in a distributed simulation form an Application Layer protocol. This protocol makes direct use of the Ethernet's Network Layer service with null intermediate layers.

Long-Haul Links

The sites are currently interconnected using terrestrial data links, each having a capacity of 56 kilobits per second. Such links are relatively inexpensive and can be readily installed to serve locations anywhere in the United States.

A small gateway computer forms the interface between a local area network and the long–haul links that connect it to other sites. In its simplest form, the gateway merely copies messages between the local area network and the long–haul links. Any message produced by a local simulator will be distributed to remote sites by the gateway, and any message received from a remote site will be distributed to local simulators via the local area network. Effectively, the collection of local area networks and long–haul links behaves as a single, large broadcast network.

This simple form of gateway can support about twelve ground vehicle simulators in a distributed simulation spanning a 56 kilobit per second link. Using more sophisticated algorithms, these links can support up to 50 vehicles. Additional vehicles can be supported by using links of higher capacity (T1 links of 1.544 megabits per second are commercially available), or by using multiple links in parallel to connect pairs of sites.

DISTRIBUTED SIMULATION APPROACH

The concept of distributed simulation is central to our networking approach. There is no central computer that directs the activities of the various simulation elements. Instead, each simulator has its own microcomputer, which is in continuous communication with each of the other simulation elements. Each simulator is responsible for dispatching messages to the other simulators to convey the information they need to know about its actions. Conversely, each simulator is responsible for receiving, interpreting, and responding properly to messages received from other simulators.

One significant advantage of this distributed simulation

approach is that as the simulation network is expanded, each new simulator brings with it all of the computational resources necessary to support itself. This means that adding new simulators does not (generally) involve hardware modifications to simulators already on the network.

Dead Reckoning

Each simulator is, of course, responsible for maintaining a detailed model of its own state including, for example, engine power, thrust, and fuel consumption; aerodynamic forces or terrain forces; weapon system computers, etc. Each simulator also maintains a simple dead reckoning model of the state of every other simulator on the network that is within possible interaction range. In essence, this involves extrapolating the last reported position of each other vehicle, based on its last reported velocity vector, until such time as a new state update message is received.

This approach implies that each simulator is also responsible for transmitting state update messages whenever it changes course or speed. To do this, each simulator must maintain, in addition to its high fidelity model, a dead reckoning model that corresponds to the model that other simulators are maintaining of its state. In essence, after each update of its high fidelity model, the simulator compares its exact state with that of the dead reckoning model and transmits a state update message only when a significant discrepancy has accumulated.

The state update message contains the essential "externally visible" information that the other simulators will need to depict the broadcast vehicle accurately on their screens: the vehicle's location and orientation, with six degrees of freedom (or more, if it has an independently movable turret, gun tube, or other significant features that can be seen at a distance), its velocity vector components, and whether it is currently producing smoke, a dust column, a muzzle blast, or other significant visual effects.

These algorithms produce a variable update rate that will differ from one simulator to another at any given time. Each simulator transmits state update information only when necessary. The principal motivation is, of course, to minimize network message traffic and hence the amount of incoming information that other simulators must process. We are currently using a maximum rate of 15 updates per second and a minimum rate of one update every 5 seconds.

Hit and Damage Determination

Another illustration of distributed simulation occurs in the shared responsibility for hit and damage determination for ballistic or missile attacks. Briefly, the firing vehicle is responsible for determining what, if anything, was hit; the target vehicle is responsible for determining what damage, if any, it suffered.

Consider the sequence of events that occurs when the gunner in vehicle A fires at vehicle B. First, the simulation host computer orders the appropriate gunfire sound effects in simulator A. It then dispatches a message to all simulators announcing the position of its muzzle blast. Next, the simulation host informs the CIG host in its own simulator of the type of ammunition fired and its initial velocity vector.

Within the CIG computer, precomputed ballistic trajectories have been stored in the form of chord segments, divided into single–frame–time intervals. As the CIG system processes the various polygons that make up the terrain and the fixed and moving objects on the terrain, it determines, on a frame–by–frame basis, whether the ballistic chord intersected any polygon. If so, it informs the simulation host which polygon was hit, and displays the approriate visual effect at the point of impact. The effect displayed depends on what kind of polygon was hit as well as the type of munition. For example, one effect is used for a shell exploding on armor, while another is used for a shell exploding in dirt.

The simulator then transmits an Impact Event message that informs all other simulators of the type and location of the impact, so that they can display the appropriate effect on their screens. For all vehicles other than the one struck by the projectile, this is the extent of the processing required. The struck vehicle, however, is now responsible for determining what damage, if any, it has suffered. To do so, it employs probabalistic damage tables that depend on additional variables contained in the Impact Event message, including the type of ammunition, the range to the target, the angle of incidence, and the precise location of the hit on the struck vehicle. In some cases, the tables depend on information known only to the struck vehicle simulator, such as (in the case of the M1 tank) whether the ammunition ready rack blast doors were open at the time of impact. If so, the probability of a catastrophic kill is substantially increased. If the struck vehicle catches fire, it broadcasts a state update message announcing that fact, so that other simulators can appropriately display it.

Self–Healing Nature of the Protocol

The state update protocol does not require the acknowledgement of each update. To do so would substantially increase the volume of network traffic, since state updates represent 95 percent of the total traffic volume in typical exercises. Instead, the updates are designed to be self–healing, in the sense that if a simulator somehow misses a state update message, the worst that will happen is that it will continue to extrapolate the previous dead reckoning information for a little while longer. As soon as the next state update is received, which could be the next frame interval for a rapidly maneuvering vehicle, the location and orientation of the vehicle will be corrected and a new extrapolation process begun. For other types of messages, multiple retries may be required if the message is not positively acknowledged. But, the types of messages for which this approach is required represent a small fraction of the total traffic, so the impact of such retransmissions is negligible. It should be noted that

these considerations are largely theoretical, since the missed packet rate observed to date has been essentially zero.

Balancing Communication, Computation, Precision

In essence, dead reckoning achieves a trade–off among three factors: the network communication traffic, the amount of computation performed by simulators, and the precision with which each simulator perceives the vehicles of other simulators. Network traffic is reduced by dead reckoning because fewer state updates are broadcast. Computation demands are increased for the simulators that must, as a result, extrapolate the appearances of vehicles in the absence of any state updates describing them. Precision is limited by the amount of discrepancy allowed to accumulate between a vehicle's high fidelity model and its dead reckoning model.

There are many parameters of the dead reckoning algorithm that may be adjusted to establish the point at which these three factors are balanced. The thresholds against which discrepancies are gauged must be carefully chosen, because as these thresholds are increased network traffic is reduced; but so is precision. There are also choices to be made among dead reckoning algorithms. Dead reckoning can be based on the use of higher order time derivatives of vehicle motion—such as acceleration—with the result that network traffic is reduced; but, more computations must be performed to extrapolate a vehicle's state using these higher order derivatives.

The optimal choice of discrepancy thresholds and dead reckoning algorithms depends on the type of vehicle simulated. The choices that are appropriate for slow moving ground vehicles are not optimal for high–speed aircraft. For this reason, we have been investigating how different thresholds and algorithms perform for different types of vehicles. Our studies are based both on empirical data—collected from actual vehicle simulators operated by soldiers in simulated battle conditions—and on theoretical analysis of the trade–offs between network traffic and computation. In the following paragraphs, we explain how these studies are conducted.

We begin our study of discrepancy thresholds and dead reckoning algorithms for a particular type of vehicle simulator by observing the way the vehicle is used by pilots, drivers, and crews. The distributed simulation approach provides a convenient method for doing this. The simulator is connected to the network and made to send state updates at its maximum rate (typically 15 per second). As crewmembers operate the simulator in a variety of battle conditions, we record the state updates originating from the simulator using a data collection system attached to the network. The record of time–stamped state update messages we thus obtain forms a complete record of how the simulated vehicle was operated by its crew.

We then filter this complete record of state update messages through a program that applies any dead reckoning algorithm and discrepancy thresholds we wish to hypothesize. The output of the program is a reduced set of state update messages, representing those less frequent updates that would actually be sent by the simulator had it been using the algorithm and thresholds hypothesized. We can measure this reduced set to determine the network traffic that the algorithm and thresholds entail. We can also replay the reduced set of messages onto the network to observe the dead reckoned model of the original vehicle and judge fidelity of that model.

Some of the data we derived in this way for M1 Abrams Main Battle Tank simulators are plotted in Figure 1. The graph shows how the network traffic produced by an M1 simulator during a five minute battle sequence is influenced by dead reckoning thresholds. The vertical axis of the graph represents the average number of state updates per second produced by the simulator for a particular set of thresholds. The three plotted lines correspond to thresholds on the vehicle's orientation of 2.5, 5, and 20 degrees—the amount by which the vehicle's actual orientation is allowed to diverge from its dead reckoned orientation before a state update must be sent. The horizontal axis represents the threshold on the vehicle's location, measured as a fraction of the vehicle's own size. A dead reckoning algorithm based on velocity was hypothesized for the simulator. The graph confirms one's intuition that increasing a discrepancy threshold will reduce network traffic. (For ground vehicles such as the M1, however, this reduction is not substantial.)

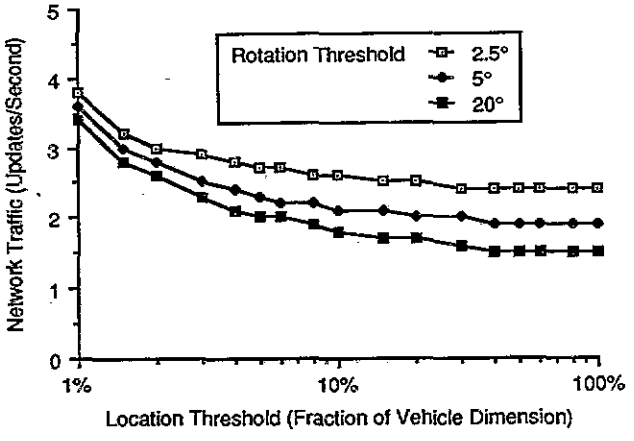

Figure 1. The frequency with which a simulator broadcasts state updates depends on both the dead reckoning algorithm and the discrepancy thresholds used. This graph shows the effect various rotation and location thresholds have on the frequency of updates from an M1 simulator employing velocity–based dead reckoning. Good performance is achieved at thresholds of 3° and 10%, resulting in 2 to 3 state updates per second.

Similar graphs are produced to characterize other dead reckoning algorithms, including those that incorporate higher–order derivatives such as acceleration. Each algorithm is analyzed to determine its computational requirements, and those requirements are weighed against the network traffic produced by the algorithm. The computational requirements must be evaluated as they affect all simulators on the network,

since all will be required to dead reckon the vehicle under consideration.

For the M1 simulator, we concluded that a velocity–based dead reckoning algorithm was most appropriate for a population of several hundred simulators supported by an Ethernet local area network. We found that network traffic was not substantially reduced when dead reckoning was allowed to take into account higher–order derivatives of vehicle motion. The discrepancy thresholds that strike a good balance between network traffic and precision of vehicle appearance, we found, are 3 degrees of rotation and 10% of the vehicle's dimension. As Figure 1 shows, this choice results in network traffic averaging 2 to 3 state update messages per second per vehicle.

In studying a close–support aircraft simulator, however, we found that significant reductions in network traffic could be obtained by using higher–order derivatives for dead reckoning. If dead reckoning is based solely on the velocity of the aircraft (updating the aircraft's position by integrating its velocity), then about six state update messages per second are needed for a reasonably precise appearance. However, if rate of rotation and linear acceleration are taken into consideration by the dead reckoning algorithm and the vehicle's velocity vector is also rotated with each update of its orientation, then only about two state update messages per second are needed. Figure 2 shows the network traffic produced by the aircraft simulator using the more elaborate dead reckoning algorithm.

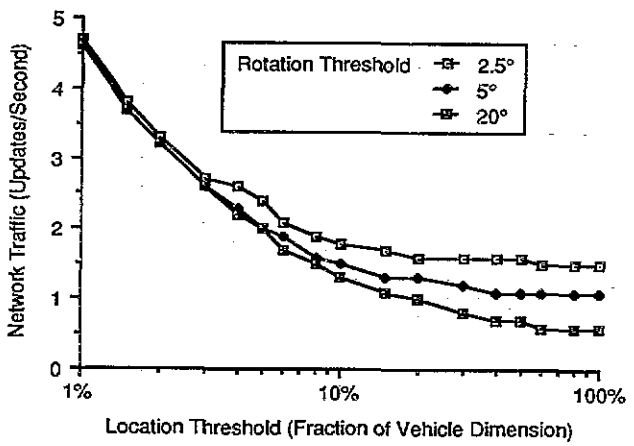

Figure 2. By using a more sophisticated dead reckoning algorithm, a close–support aircraft simulator can be made to produce even fewer state updates than an M1 simulator. For this graph, the aircraft simulator's dead reckoning employed velocity, linear acceleration, and rate of rotation information. At thresholds of 3° and 10%, only 1 to 2 state updates are produced per second.

One can see by comparing Figures 1 and 2 that, with the appropriate choices of dead reckoning algorithms, the aircraft simulator can be made to produce less network traffic than the M1 simulator. This may be so because the motion of the simulated aircraft is influenced primarily by its pilot, whereas that of the tank is influenced also by its interaction with the ground. At the scale at which the discrepancy thresholds apply, the tank's motion is more erratic than the aircraft's motion because of that interaction with the ground. Thus dead reckoning is able to predict the motion of the tank over less extended periods of time.

Future Extensions of Long-Haul Networking

We have described a trade–off involving the communications bandwidth required to convey the state update messages produced in a distributed simulation and the processing power required to extrapolate a vehicle's appearance in the absence of an update describing it. For a given level of precision, an optimal position can be found that balances the cost of local area network bandwidth with that of simulator processing power. A different balance is dictated, however, by the comparative costs of long–haul link bandwidth and gateway processing power. Because gateways are relatively few, and long–haul links are relatively expensive, it may make sense to invest more processing in the gateways to reduce the need for long–haul link bandwidth.

We are exploring two ways in which the limited bandwidth of long–haul links can be used more efficiently, albeit at the expense of greater gateway processing. One way is to reduce the frequency of state update messages by employing dead reckoning algorithms based on higher–order derivatives of vehicle motion. The other way is to compress the information in a state update message so that fewer bits are required per message.

Our use of dead reckoning can be viewed as a way of abstracting information about the appearance of a vehicle so that the appearance need be conveyed less often. As we demonstrated in the previous section, it is possible in some cases to achieve greater levels of abstraction by using higher–order derivatives of a vehicle's motion. The costs of greater abstraction are measured in the processing required to first construct the abstraction, and that required to later interpret it. We are investigating ways that gateways may perform this processing to reduce the number of state update messages transmitted across long–haul links without increasing the processing burden of individual simulators.

Data compression can also be used to increase the number of vehicles that can be simulated within any given long–haul link bandwidth. Significant portions of a state update message are relatively static. Of course, these are the provisions in the protocol that allow it to be self–healing. However, given that the communication between two gateways is more controllable than that on the broadcast Ethernet, we can remove this redundant information and represent messages with fewer bits.

Conclusion

For the first time, multiple ground vehicle and aircraft simulators have been joined on a common, virtual battlefield

spanning multiple physical sites. This low–cost, low–bandwidth simulator network is now in daily use for unit training and experimental development. The development and continued refinement of algorithms for reducing effective bandwidth requirements allow ever–greater numbers of simulators to share the network. We expect that in years to come, these techniques will become standard, and that units will regularly engage in combined arms training without leaving their home posts or bases.

ABOUT THE AUTHORS

Dr. Miller is Divisional Vice President of the Simulation and Training Systems Division of BBN Systems and Technologies Corporation. He holds an Sc.D. in Mechanical Engineering from M.I.T., specializing in control theory and man–machine system interaction. He joined BBN in 1963.

Mr. Pope leads the Command and Control System Simulation group of BBN's Simulation and Training Systems Division. Since joining BBN in 1983, he has also been involved in the development of standards for the interchange of electronic mail. He has a BS degree in Computer Science from the University of British Columbia.

Mr. Waters is responsible for long-haul network communications within BBN's Simulation and Training Systems Division. Since joining BBN in 1984, he has also been involved with the development of packet speech applications within DARPA's Wideband Packet Satellite Network. He has BS degrees in Computer Science and Electrical Engineering, and an MS in Computer Science from Washington University in St. Louis.

REFERENCES

[1] James Chung, Alan Dickens, Brian O'Toole, and Carol Chiang. *SIMNET M1 Abrams Main Battle Tank Simulation: Software Description and Documentation (Rev. 1).* BBN Systems and Technologies Corp. Cambridge, Mass., Aug. 1988.

[2] James Chung, Alan Dickens, Brian O'Toole, Carol Chiang, Warren Katz, and Bryant Collard. *SIMNET M2/3 Bradley Fighting Vehicle Simulation: Software Description and Documentation.* BBN Systems and Technologies Corp. Cambridge, Mass., Aug. 1988.

[3] Dan Friedman and Varda Haimo. *SIMNET Network Performance.* BBN Report Number 6711. BBN Communications Corp. Cambridge, Mass., Jan. 1988.

[4] Arthur Pope and Duncan Miller. "The SIMNET Communications Protocol for Distributed Simulation." *Proceedings of the Sixth Annual Technology in Training and Education (TITE) Conference.* Biloxi, Miss., March 1988.

[5] Arthur Pope. *The SIMNET Network and Protocols.* BBN Report Number 6787. BBN Laboratories, Inc. Cambridge, Mass., May 1988.

[6] Lt. Col. Jack A. Thorpe, USAF. "The New Technology of Large Scale Simulator Networking: Implications for Mastering the Art of Warfighting." *Proceedings of the Ninth Interservice/Industry Training Systems Conference,* pp. 492–501. Washington, D.C., Nov. 1987.