Time Management in Distributed Simulation Models

T. Michael Baker, PhD
Adacel Technologies Limited
2 Portrush Road
PAYNEHAM SA 5070

Keywords:
Time, Time Management, Distributed, Simulation, Models, DICE, dGAME, IASSF, DIS, ALSP, HLA

ABSTRACT: There are several methods by which time can be managed in a simulation model. Problems can arise in time management when a simulation model is distributed over two or more machines.

The purpose of any time management scheme should be to ensure temporal causality among simulated events. Time management in a distributed simulation can be based on:

- using synchronised system clocks,
- sending a time sync signal from a central clock,
- dividing time into discrete periods and not proceeding to the next time period until all calculations for the current time are complete,
- tying a discrete time simulation to a clock, and
- several hybrids.

Methods of time management for single distributed simulations can be based on those used in multiple simulation architectures, such as Aggregate Level Simulation Protocol and High Level Architecture. Aggregate Level Simulation Protocol uses a conservative time management scheme, whereas High Level Architecture provides support for several different time management schemes.

1. Introduction

This paper covers how time can be dealt with in distributed simulations. There are several approaches that can be used. This paper includes examples of several of these approaches taken from the author's experience.

Methods by which time can be managed in a simulation model include time-stepped, event driven and independent time advance. In a time-stepped simulation, time is advanced in steps of a pre-determined length. In an event driven simulation time is advanced to the next event. In independent time advance simulations, time advances independently of the events occurring in the simulation. Usually in such simulations time advances in line with wall clock time or as a multiple of wall clock time, either faster or slower.

Problems that can arise in time management when a simulation model is distributed over two or more machines include network latency causing temporal anomalies and variability causing non-repeatability. A temporal anomaly is illustrated in Figure 1. In this illustration a simulation is being run on three processors. An event (A_1) simulated on Process 1 is transmitted to Processes 2 and 3. Due to network latency notification of the event (A_2 and A_3) to Processes 2 and 3 is delayed. As a result of the event (A_2) Process 2 simulates a second event (B_2). This is transmitted to Process 3 but arrives before the first event (A_3).

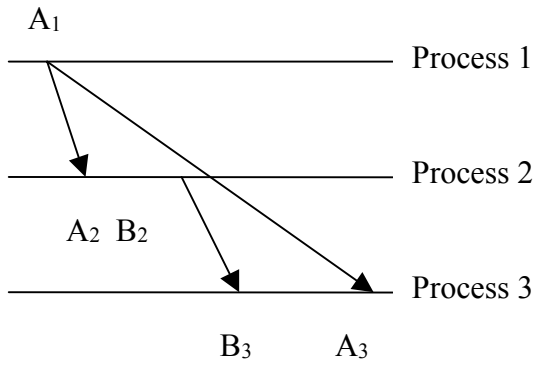

Figure 1: Temporal anomaly

1.1 Types of Simulation

Simulations can be divided into several types, both by their purpose and by the time management approach that they employ. Three of the main purposes for simulations are Training, Analysis, and Test and Evaluation. Four of the main time management approaches for simulation are Real Time, Time Stepped, Event Driven, and Optimistic Time Warp. In Optimistic Time Warp calculations are made ahead of the current simulation time, and if events arrive in the calculations past, then the calculations are wound back to the time of the event that arrived.

For most types of analysis two time aspects are important. Any analysis scenario run on a simulator should be repeatable, and should run as fast as possible.

Some approaches to time management lead to minor time anomalies, such as a weapon firing after it has hit its target. If time anomalies are caused by network latency, then they will occur randomly and scenarios will not be repeatable. Non-repeatability is a problem when a scenario is re-run with some change made. If the results of the two scenarios are different it is very difficult or impossible to tell if the difference is due to the change or due to the non-repeatability.

1.2 Importance of Time for Analysis

Time is especially important in distributed simulations used for analysis because of causality and repeatability.

In a simulation running on a single process time is easy to deal with. Regardless of how time is dealt with, a single process can ensure that events occur in a logical sequence corresponding to real events being simulated.

For a simulation running in multiple processes time is not a problem if the processes are running synchronously. That is if a mechanism exists to ensure that each process advances time in sync with every other process.

However there can be a problem if multiple simulation processes advance time asynchronously. Under such circumstances an event simulated by one process can occur in the past of part of the simulation running in another process.

Further complications can arise when multiple simulation processes are running on separate processors that are connected by a network. In this case there is the further problem of network latency to deal with.

1.3 Time is Less Important for Training

The time management requirements for Training and for Test and Evaluation are a lot less stringent because in general there is no need for repeatability. Also minor time anomalies, particularly if of a very short time period, are either difficult for a human to detect, or have minimal impact on human players in interactive simulations.

1.4 Problems in Distributed Simulation

The two major problems that can occur with respect to time in a distributed simulation are time anomalies and non-repeatability.

While time anomalies may be minor in a real time simulation, they can become a major problem in faster than real time simulations. This is because the simulation rate or time multiplier expands the size of the time anomaly. For example it could lead to a simulated aircraft being destroyed before it detects a missile and has an opportunity to take evasive action.

If the message delivery latency that causes time anomalies is variable then it will also lead to potential non-repeatability. For example if messages arrive in different orders in two runs of the same scenario, the outcomes may be different.

2. Examples

The following examples illustrate several methods that can be used the manage time in single simulations that are distributed over several processors.

The Distributed Interactive C3I Effectiveness (DICE) simulation uses scaled wall clock time and requires that the system clocks of all processors supporting a simulation be synchronised.

The Distributed Generic Agent Modelling Environment (dGAME) uses a conservative time management scheme in which models bid for time advances and the smallest bid is granted.

The Integrated Avionics System Support Facility (IASSF) contains avionics simulations that are effectively time stepped, in real time, by the MIL-STD-1553 bus.

2.1 DICE

DICE was designed by the Defence Science and Technology Organisation (DSTO) and has been implemented by Adacel Technologies Limited to model and simulate Command, Control, Communication and Intelligence (C3I) systems for effectiveness studies, training and mission rehearsal [1].

The DICE simulation has general-purpose applicability and consists of artificial agents which complement optional human players, and a means of interfacing to battlefield simulations and other models that represent the overall military mission, operation or battle. The impact of C3I aspects on an overall mission can be used to gauge C3I system effectiveness. The DICE simulation is designed to facilitate inter-operation with in-service or operational command support systems.

While DICE can be run on a single Windows 95 PC, optimal performance is achieved when it is run on a network of Windows NT and Sun Solaris machines. DICE models the communications between a number of agents. Text based messages are passed between agents over communications links. Each communication link can have a delay which models the time taken between the decision to send a message and the time at which the message is received and understood.

The initial design was for a timeserver to periodically send simulation time messages to each agent, and for each agent to keep its own clock in sync with all other agents' clocks.

It was apparent that, because of variation in network latency, there would be considerable variation between agent's clocks. These variations could have been smoothed out statistically. However as this is the method used by the Simple Network Time Protocol [2] (SNTP) it was decided that it would be easier to use SNTP to keep each system's clock in sync and calculate simulation time from scaled system clock time.

Each machine that is participating in a DICE simulation has its system clock synchronised with each other machine in the simulation using the SNTP. The current simulation time of a DICE simulation is maintained relative to the current system time of the hosts on which the simulation is executing. The current simulation time is calculated from the system time by

adding an offset and multiplying by a (possibly fractional) factor. The offset and multiplication factor are identical for each node in the simulation. The system time used is Universal Coordinated Time (UTC) so machines from different time zones can participate in the simulation with no special action required.

Agents are built on top of a Common Node Framework (CNF). The CNF delivers data from incoming messages to the agent and constructs outgoing messages from data supplied by the agent. The CNF spends most of its time waiting for incoming messages. On receipt of a message and delivery of the data from the message to the agent, or after a message wait timeout period has elapsed, the CNF calls the agent's step function. When the CNF calls the step function it passes the agent the current simulation time, which it calculates from the system time. As the agent is only active when processing its step function, the agent is able to give the CNF a simulation time at which it must next be activated. The CNF uses any such activation time when determining its next wait for incoming message timeout.

Given the structure of DICE, the potential exists for time anomalies to occur. However, if care is taken in assigning agents to processors, the likelyhood of their occurring is small. Messages are sent over links that have simulated link delays. The delay is simulated at the receiving end of the link by that agent's CNF putting it into a queue until the delivery time. The minimum delay used on a link is likely to be one second. Provided all agents with links between them with these small delays are run on the same Sun machine, the scaled network latency, when running at ten times normal speed, should be no more than about 100 milliseconds. This should ensure that all messages are waiting in the agent's CNF queue before they are due to be delivered.

2.2 dGAME

dGAME provides an environment in which an analyst can assemble physical and/or reasoning models and run them in a scenario [3]. The Australian Artificial Intelligence Institute is developing dGAME with assistance from Adacel Technologies Limited for the Defence Science and Technology Organisation.

A dGAME scenario consists of a number of interacting entities. Each entity is connected to a model that models its physical or reasoning behaviour. Each entity can publish data on data busses. To minimise the amount of data transferred, other entities can subscribe to, and unsubscribe from, individual data busses.

The model attached to an entity can be changed during a scenario to reduce computational load. For example an aircraft entity might be connected to a point mass model for straight and level flight, and to a six degree of freedom model for manoeuvring.

Models are assembled in model servers that can be distributed over several processors. dGAME provides for the transfer of data between servers as required.

dGAME is initially being developed to run on networked multi processor SGI machines running IRIX. However one of the design aims is that it be portable to other architectures.

Given that the principle use of dGAME will be for analysis, one of the main design constraints is that scenarios must be repeatable.

After a round of calculations in a dGAME simulation each model makes a time advance request. The simulation time is then advanced by the amount of the smallest request made.

dGAME provides no checks that there are no cyclic dependencies between entity attributes that will cause a scenario to lock up.

If a dGAME scenario that normally runs at faster than real time is to be run with a Human in the Loop (HIL), then a time server can be added. The time server waits a short period then makes a sufficiently small time advance request so that simulation time advances are no more than the advance in wall clock time.

2.3 IASSF

IASSF will allow the RAAF to support the avionics system of the frontline F/A-18 fighter in Australia. GEC Marconi Systems developed IASSF for the Royal Australian Air Force (RAAF).

The capabilities of the IASSF include the stimulation and simulation of all the aircraft's major avionics sub-systems, such as the Mission Computer, Communications, Radar, Stores Management, Electronic Warfare and Navigation systems. The facility can also integrate these simulated systems with real aircraft avionics. The RAAF will be able to select combinations of real or simulated avionics systems to allow it to conduct controlled testing of any future operational flight programs or avionics upgrades [4].

The avionics in the F/A-18 work on a MIL-STD-1553 multiplexing (MUX) data bus. Each bus has a Bus Controller (BC) and one or more Remote Terminals (RTs) attached to it. Each of the F/A-18's sensors (Radar, Electronic Warfare and Forward Looking InfraRed) are RTs. The bus runs at a frequency of 20 Hz. That is, every 50 ms the BC sends data to the RTs, which then return data to the BC.

The design of the MUX bus imposes the time regime on all of the RT avionics models within IASSF. The IASSF sensor models, like the other IASSF avionics models, are derived from a generic RT model. The generic model handles the receipt and dispatch of messages from and to the MUX bus. The derived models provide a function that is called once per MUX bus cycle after the receipt of incoming messages and before the dispatch of outgoing messages to update the state of the model. As a consequence the simulation models work in real time, updating their state every 50 ms.

IASSF simulation models provide repeatable results provided they don't overrun their allotted processing times. If they do overrun then, like MIL-STD-1553 connected avionics that overrun their allotted processing times, they don't return messages and miss the next calculation cycle. If the time overrun is due to some transient condition like an unrelated process being run on the model's processor, then this will lead to non-repeatable results. However given that the models run on processors that only run models, this would be unlikely to occur.

The models run on a number of processors, each of which are connected to MUX busses, as well as a LAN and a shared memory bus.

3. Standards

Several standards have been developed that cover distributed simulations. These include Distributed Interactive Simulation (DIS), Aggregate Level Simulation Protocol (ALSP), and High Level Architecture (HLA). Each standard has its own method or methods of dealing with time.

DIS provides no time coordination between the independently running real-time simulators participating in a DIS simulation.

Time management in ALSP consists of simulation time in each connected simulation being coordinated so that event causality is maintained.

The HLA Run Time Infrastructure (RTI) provides sufficient time management functions so that Real Time, Time Stepped, Event Driven, and Optimistic Time Warp simulations can all run in the same confederation.

3.1 DIS

DIS connects arms training and other simulation systems so that they can participate in the same simulation exercise, which allows team training to take place. The concept of networked simulation was first developed in the SIMNET program, sponsored by the Defense Advanced Research Projects Agency (DARPA). DIS extended the concept of SIMNET to networking simulators of differing manufacture and different fidelity and function [5].

Interaction between simulations is achieved by the broadcasting of Protocol Data Units (PDUs), each of which contains a packet of data describing an event in the simulation. To reduce network traffic, entity states in a DIS simulation are dead reckoned. PDUs are broadcast using the unreliable UDP/IP, so PDUs may be lost [6]. However the effect of lost PDUs is minimal because of the use of dead reckoning.

Each PDU contains a timestamp indicating when the PDU was issued. Entity PDUs contain dead reckoning information that other simulations use to keep track of where the entity is. The simulation that owns an entity uses both a high fidelity model and a dead reckoning model. When the difference between the two models exceeds a threshold the simulation issues an updated entity PDU.

The command to start or resume a DIS simulation contains both the simulation time and the actual time at which the simulation is to start [7]. DIS simulation time is the reference time (e.g., Universal Coordinated Time) within a simulation exercise. Simulation time is established ahead of time by the simulation management function and is common to all participants in a particular exercise.

3.2 ALSP

ALSP is a set of software and protocols used extensively by the United States military to inter-operate analytic and training simulations in the ALSP Joint Training Confederation (JTC) [8].

The ALSP design principles were, no central node, geographic distribution, message-based protocol, time management, data management, and attribute ownership. Unlike the real-time simulators that participate in DIS, the simulations that interoperate using ALSP are discrete event simulations that explicitly manage time. ALSP provides time management services to coordinate simulation times and preserve event causality across simulations [9].

With the replacement of DIS and ALSP by HLA, the JTC is being transitioned to use HLA [10].

3.3 HLA

The major functional components of the HLA are federates, the Run Time Infrastructure (RTI) and the interface between the federates and RTI. Federates are the simulations participating in an HLA federation. The RTI acts as a distributed operating system for the federation. Two of the services that the RTI provides are object data exchanges between federates and time management [11].

The HLA enables federates to manage local time in a way that allows them to coordinate data exchange with other members of a federation. The HLA time-management structure supports interoperability among federates using different internal time-management mechanisms. The HLA supports these capabilities provided that federates adhere to certain requirements necessary to realise each service.

To achieve these goals, a single, unifying approach to time management provides time-management interoperability among disparate federates. Different categories of simulations are special cases in this unified structure, and typically use only a subset of the RTI's full capability. Federates need not explicitly indicate to the RTI the time-flow mechanism (time stepped, event driven, independent time advance) being used within the federate, but utilise the RTI services (including time management) that are appropriate for coordination of data exchange with other federates [12].

Data sent from one federate to another can be delivered by the RTI either in delivered order or in timestamp order (TSO). For TSO the sending federate has to timestamp the outgoing data and the receiving federate has to be able to receive TSO data. The RTI ensures that TSO data is delivered in TSO.

4. Conclusions

Several approaches are available for time management in distributed simulation, that, in most cases, ensure repeatability and that time anomalies don't occur.

In general distributed simulations are simpler than the mixed federations that HLA can support. Consequently the time management for a distributed simulation can be much simpler, using a single approach instead of the many that the HLA has to support.

5. References

1. Davies, M., C. Gabrisch, J. M. Dunn and F. D. J. Bowden. The Distributed Interactive C3I Effectiveness (DICE) Simulation, DSTO-TR-0485, Information Technology Division Electronics and Surveillance research Laboratory, DSTO, Salisbury, (1997).
2. Mills, D. Simple Network Time Protocol (SNTP) Version 4 for Ipv4, Ipv6 and OSI. Network Working Group RFC2030, (1996) <http://www.faqs.org/rfcs/rfc2030.html>.
3. Evens-Greenwood, P., D. Appla, I. Lloyd, G. Murray, and G. Tidhar. dGAME - A Distributed Generic Agent Modelling Environment, In: Proceedings of SimTecT 99.
4. Australian Defence Community. Industry Leaders, GEC-Marconi, Australian Defence Community. <http://www.adc.gov.au/sponsors/GEC_4.html>.
5. Williams, J. Distributed Interactive Simulation, Institute for Simulation and Training, Orlando (1998). <http://www.ist.ucf.edu/labsproj/projects/dis.htm>.
6. Locke, J. An Introduction to the Internet Networking Environment and SIMNET/DIS. 12pp. Computer Science Department, Naval Postgraduate School, Monterey (1995).
7. IST. Standard for Distributed Interactive Simulation – Application Protocols Version 2.0 Fourth Draft (Revised). Institute for Simulation and Training, Orlando (1994).
8. Fischer, M. C. Joint Simulated Battlefield Through Aggregate Level Simulation Protocol. In Proceedings Southeastern Simulation Conference '95, Orlando, FL. (October 1995) <http://alsp.ie.org/alsp/SESC_Paper/sesc95w.html>.
9. Miller, G., and A. Zabek. The Joint Training Confederation and the Aggregate Level Simulation Protocol. Phalanx (June 1996). <http://alsp.ie.org/alsp/mors_96_miller/mors_96.html>.
10. Griffin, S. P., E. H. Page, Z. Furness, and M. C. Fischer, Providing Uninterrupted Training to the Joint Training Confederation (JTC) During Transition to the High Level Architecture (HLA). In: Proceedings SimTecT 97. <http://alsp.ie.org/alsp/SIMTECT97/PAPER.html>.
11. Dahmann, J. Evolution of DoD M&S Strategy. Defense Modeling & Simulation Office, Alexandria, VA (1998) <http://hla.dmso.mil/hla/general/annotate/index.htm>.
12. Simulation Interoperability Standards Organization (SISO) Standards Development Group (SDG). Draft Standard [for] Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules, IEEE P1516/D1. Institute of Electrical and Electronics Engineers, Inc., New York (1998).

6. Author Biography

Dr Michael Baker is a Consultant Systems and Software Engineer with Adacel Technologies Limited overseeing the development of DICE and dGAME.

Dr Baker obtained a BSc (1st Hons.) in Civil Engineering in 1972 and MPhil in Town and Regional Planning in 1974 from the University of Glasgow and PhD in Energy Research in 1981 from The Open University.

In the Academic Computing Service at The Open University, Dr Baker developed a model of UK energy production and consumption for use by students in developing UK energy scenarios and a model of house energy use for giving consumers energy conservation advice.

While acting as an independent software consultant Dr Baker developed an Animated Interface for Computer Based Training.

More recently Dr Baker was responsible for the design and implementation of sensor interface simulation models for the IASSF.