Modelling of Dead Reckoning and Heartbeat Update Mechanisms in Distributed Interactive Simulation

Peter Ryan

Will Oliver

Air Operations Division
Defence Science & Technology Organisation (DSTO)
506 Lorimer St, Fishermans Bend, Melbourne, Victoria, 3001, AUSTRALIA
Peter.Ryan@dsto.defence.gov.au

Keywords:
Distributed Interactive Simulation; Dead Reckoning; Heartbeat

ABSTRACT: *Dead reckoning is employed in Advanced Distributed Simulation exercises to reduce the need to continually update a simulated entity's state information. The IEEE Distributed Interactive Simulation (DIS) protocol provides a standard set of 11 algorithms for entity position and orientation dead reckoning. Related to dead reckoning is the heartbeat mechanism used to periodically define the current state of an entity. The IEEE standard defines heartbeat rates for many Protocol Data Units (PDUs) with default values specified for rate of issuance of heartbeat PDUs and timeout. Both the dead reckoning and heartbeat mechanisms are required to maintain continuity of entities within a distributed simulation exercise. To study dead reckoning and heartbeat, a model was developed that simulates these two mechanisms within DIS for entity position. This model can perform sensitivity analysis on the critical parameters, such as entity speed, heartbeat rate and dead reckoning threshold for different dead reckoning algorithms. The model demonstrated that the second order algorithm using acceleration and velocity is far more efficient than the first order algorithm that used only velocity. Third order algorithms that use rate of change of acceleration were also studied and shown to offer only marginal improvement while requiring additional computational resources. The model also demonstrated that the default values for threshold are too small for fast moving entities such as aircraft and can be used to determine more appropriate values. The fraction of heartbeat to dead reckoning PDUs was examined for several different entity types (tank, aircraft, missile) with appropriate settings for threshold and heartbeat rate. It was shown that the use of a fixed heartbeat rate is not appropriate for all entity types and that heartbeat PDUs predominate for high values of threshold. The model also shows that issuance of heartbeat PDUs depends on which dead reckoning algorithm is used: if an efficient algorithm that requires few dead reckoning PDUs is adopted, more heartbeat PDUs are then required to maintain the entity within the simulation exercise.*

1. Introduction

Advanced Distributed Simulation (ADS) was created to link simulators, simulations and/or real devices so that the various entities can interact with each other to conduct a simulated game or exercise in the same synthetic battlespace. ADS has been under development since the early 1980s with the Simulator Networking (SIMNET) Project undertaken by the US Defense Advanced Research Projects Agency [1] and has continued through the emergence of Distributed Interactive Simulation (DIS) [2] in the early 1990s and High Level Architecture (HLA) [2] in the late 1990s. In parallel with these efforts, the Test and Training Enabling Architecture (TENA) has been established to enable the live range community to participate in distributed simulation exercises [3].

Distributed Interactive Simulation (DIS) is a networking protocol standard that provides a method of communicating entity state and other information such as voice communications, radar and sonar emissions through Protocol Data Units (PDUs). These PDUs consist of data packets which are broadcast over the simulation network. Standards for DIS PDUs were developed under the guidance of the DIS Coordinating Committee based in the US and utilising the Institute of Electrical and Electronic Engineers (IEEE) Standards approval process [5], [6], [7]. The latest standard IEEE-1278.1a-1998 was released in 1998 [7].

A key feature of DIS is its use of a technique known as dead reckoning to limit the rate at which simulated entities need to update their entity attributes such as position and velocity. DIS also uses the so-called heartbeat mechanism to maintain the continuity of entities within a simulation exercise when no dead reckoning PDUs are issued. This paper describes a model that simulates the dead reckoning and related heartbeat mechanisms in DIS.

2. Dead Reckoning and Heartbeat Mechanisms in DIS

DIS employs both dead reckoning and heartbeat mechanisms are required to maintain continuity of entities within a distributed simulation exercise. These mechanisms are described in the following sections.

2.1 Dead Reckoning Mechanism

DIS employs dead reckoning to limit the rate at which Entity State PDUs (that include the entity's positional and orientation information) are issued. By estimating the position/orientation of other simulated entities, it is not necessary to receive a report about every change in position/orientation that occurs in the entities trajectory over time. Only when a change in position and orientation differs by a prespecified amount (threshold) from the dead reckoned position and orientation is a new update required.

Further, each simulation application maintains both a high fidelity model of the position and orientation of the entities it generates (such as aircraft) and also a dead reckoned model. When these differ by a given threshold amount, an Entity State PDU is issued by the simulation application to inform the other participating simulations. Smoothing techniques can be used to eliminate jumps that may occur in a visual display when the dead reckoned position/orientation of an entity is corrected to the most recently communicated position/orientation.

Dead reckoning can also be applied to articulated parts such as a tank turret or submarine periscope. Absolute linear error for each articulated part in a chain of connected parts is used to determine if a threshold has been exceeded.

2.2 Heartbeat Mechanism

Related to dead reckoning is the heartbeat mechanism used to periodically define the current state of an entity. Both the dead reckoning and heartbeat mechanisms are required to maintain continuity of entities within a distributed simulation exercise. For example, if an entity is travelling in a straight line at constant velocity, then its position can be accurately extrapolated at any future time so that no dead reckoning update PDUs will be issued. Thus, if the heartbeat PDUs were not issued, the entity would time out and disappear from the simulation.

The IEEE standard defines heartbeat rates for many PDUs with default values specified. These include simulation for radios via the Transmitter PDU and IFF systems via the IFF PDU (implemented in IEEE-1278.1a-1998). However, there is some flexibility in the standard. The values in Table 1 are the default values defined by the IEEE standard.

Table 1: Default heartbeats for various PDUs as defined by IEEE standard

PDU Type	Heartbeat Rate
Entity State and Entity State Update	5 s
Transmitter (static/moving)	5 s / 2 s
Receiver	5 s
Designator	5 s
Electromagnetic Emission	5 s
IFF/ATC/NAVAIDS	10 s

The Entity State heartbeat rate uses the symbolic name HRT_BEAT_TIMER that has a default value of 5 s. An entity times out of a simulation if an update PDU is not received within a time interval determined by multiplying the heartbeat multiplier rate HRT_BEAT_MPLIER (with default value of 2.4) by the heartbeat rate – giving a default value of 12 s.

Other PDUs also use the heartbeat mechanism including Underwater Acoustics, Aggregate State, IsGroupOf, Minefield State, Environmental Process, Gridded Data, and Time Space Position Information [7].

2.3 Dead Reckoning Algorithms

DIS provides a standard set of 11 dead reckoning algorithms. These include expressions for both positional and orientation dead reckoning. They are specified by 3 letters – the first indicates whether the entity is fixed (F) or rotating (R), the second indicates whether dead reckoning rates are held constant as either rate of position (P) or rate of velocity (V), while the third indicates whether world coordinates (W) or body axis (B) coordinates are used. Thus FPW specifies an algorithm with orientation fixed and a constant rate of position in world coordinates [6] and is given by:

$$P_1 = P_0 + V_0 \Delta t$$

The second order algorithm uses both the velocity and acceleration terms and is defined as:

$$P_1 = P_0 + V_0 \Delta t + \tfrac{1}{2} A_0 (\Delta t)^2$$

which is the Fixed Velocity World (FVW) algorithm. Here subscript "0" refers to the parameters position (P), velocity (V), and acceleration (A) at the start of the interval and subscript "1" refers to the same parameters

calculated at the end of the interval of duration Δt. Thus the first order algorithm contains only the velocity term, whereas the second order algorithm includes the acceleration term. These values are calculated at the start of the interval.

2.4 Threshold Calculation

The DIS IEEE 1278 standard (4.5.2.1.2.1 of [6]) defines "*the default method for calculating positional accuracy as a threshold change in any direction or orientatio*n", and (B.1.1) states that "*Only when a change in position and orientation differs by a prespecifed amount (threshold) from the dead reckoned position and orientation is a new update required*." These statements are ambiguous, and can been interpreted two ways:

1. Component difference: If any of $|\Delta x|$, $|\Delta y|$, $|\Delta z|$ exceeds the threshold, where x, y, z define the positional vector, then an Entity State PDU must be issued.
2. Vector difference: If $|\Delta p|$ exceeds the threshold, where p is the positional vector, then an Entity State PDU must be issued. The formula for calculating vectorial difference is:

$$|\Delta p| = \sqrt{(x_d - x_0)^2 + (y_d - y_0)^2 + (z_d - z_0)^2}$$

Whilst the vector difference method is more complicated to calculate, it ensures that an entity is never more than 1 m (or the prescribed threshold) away from its actual location, whereas using the component difference method, an entity can be within the threshold bounds yet be further away from its actual location, for example, $\Delta x = \Delta y = \Delta z = 0.9$ m, $|\Delta p| \approx 1.56$ m.

The vector difference method has been used in the present work.

2.5 Previous Work on Dead Reckoning

Previous work has been carried out on dead reckoning. Lin and Schab [9] assessed the performance of dead reckoning using a software tool considering four factors: network load, computational load of the simulator, accuracy of the dead reckoned trajectory compared with the true trajectory, and smoothness of the dead reckoned trajectory in the visual display using both data from a flight simulator and an artificial race track trajectory. Parts of this work were also published elsewhere [10 - 11].

This work predates the initial IEEE DIS standard and appears to have been carried out to determine which dead reckoning algorithms provide the best results. In [11], for example, the authors examined many candidate algorithms to determine which gave the best performance in dead reckoning the flight simulator data. Many of these trial algorithms were not included in the final DIS standard set of dead reckoning algorithms.

Durbach and Fourneau studied the performance evaluation of dead reckoning algorithms from a network perspective [12]. PDU inter-arrival times were modelled using a two state Markov process with parameters for the model determined from experimental data. These authors did not consider dead reckoning algorithms in detail but rather developed a model to fit experimental data.

A more recent publication [13] examined the requirement for adaptive dead reckoning algorithms. These authors proposed an adaptive mechanism that sets thresholds depending on Area of Interest and Sensitive Region where update packets are only sent to relevant entities depending on threshold level, so that a close entity may receive more updates than a distant entity.

Dead reckoning is also used in distributed multi-player games to reduce network traffic. Recent work indicated the need for globally synchronized clocks to improve accuracy [14].

3. Simulation Model for Dead Reckoning and Heartbeat Update

To investigate the dead reckoning and heartbeat mechanisms in DIS, a simulation model was developed in the scripting language *tcl/tk* to study the effects for both circular and elliptic manoeuvres. The circle is the most tractable from a computational perspective and thus dead reckoning calculations are straightforward. The ellipse includes one extra parameter since it has both minor and major axes.

This model enabled sensitivity analysis to be performed for the critical parameters of speed, radius, dead reckoning threshold, and heartbeat update rate. The model computes an entity's trajectory around a circular path as perceived by another simulation. At each time interval, the dead-reckoned position is calculated using the equations in section 2.3 for both first and second order algorithms. When this calculated position deviates from the exact position by the threshold value, it is reset to the exact position to emulate the update mechanism, and the simulation continued.

In earlier work, results were also calculated for other trajectories such as an ellipse which resulted in similar findings to the present work [13].

For the case of an entity travelling in a horizontal circle, the equations of motion are simply given as:

$$x = r\cos\omega t;\ y = r\sin\omega t$$
$$\dot{x} = -r\omega\sin\omega t;\ \dot{y} = r\omega\cos\omega t$$
$$\ddot{x} = -r\omega^2\cos\omega t;\ \ddot{y} = -r\omega^2\sin\omega$$

where ω is the angular velocity, and r is the radius. The remote entity's dead reckoned trajectory is tangential to the exact circular trajectory until an update PDU is received.

Figure 1 shows a screen shot from the model.

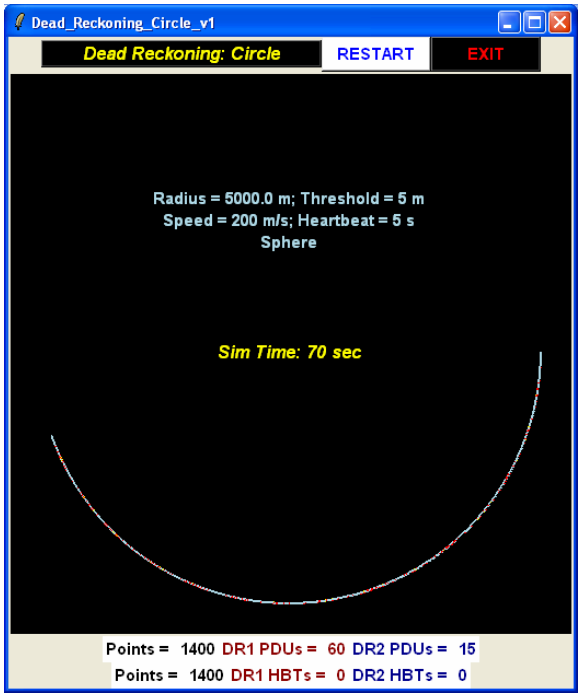

Figure 1: Screen shot from simulation model on PC

3.1 First and Second Order Algorithms for Circle

Table 2 shows the number of Entity State PDUs that would be issued for circles of different radii for a complete orbit using both the first and second order algorithms with threshold set to 1 m in all cases. The aircraft speed is set to 200 m/s and heartbeat update rate to a high value of 1000 s so that the differences between the two algorithms can be studied.

As expected, the number of PDUs issued increases as the radius is increased, although this does not scale linearly. For example, with a 20000 m radius, 628 PDUs are needed if the first order algorithm is used, whereas only 125 are needed for a circle with 1000 m radius. For smaller radii, the aircraft will be constantly turning in a tight circle thus resulting in deviation from the linear and quadratic tangential trajectories predicted by the dead reckoning algorithms, whereas for larger radii the deviation from the dead reckoning algorithms will be less marked.

Table 2: PDUs issued for different radii

Radius (m)	Number of PDUs	
	First Order (FPW)	Second Order (FVW)
1000	125	33
2500	196	46
5000	314	58
10000	418	73
20000	628	93

Further, for each radius considered, at least three times as many PDUs are required to remain within threshold for the first order algorithm, than for the second order algorithm. This indicates that the second order algorithm much more effectively approximates the entity's trajectory than does the first order.

In Table 3, the effect of varying the threshold is examined with an aircraft traveling at 200 m/s around a circle of radius 5000 m. As expected, the number of PDUs required decreases when the threshold is increased since the predictive algorithms remain within threshold for longer resulting in fewer entity updates.

Table 3: PDUs issued for different thresholds

Threshold (m)	Number of PDUs	
	First Order (FPW)	Second Order (FVW)
0.5	392	71
1	314	58
2.5	196	42
5	130	34
10	98	27
25	62	20
50	43	16

Varying both radii and threshold independently shows the expected behaviour: (a) as radius is increased, more PDUs are required, while (b) as threshold is increased, fewer PDUs are required to predict the entity's trajectory.

Of greater significance in these results is the considerable difference between the numbers of update PDUs that would be required for the two algorithms. In every case, the first order algorithm requires about three times as many update PDUs to be broadcast as the second order algorithm. Thus the second order algorithm provides a far more efficient means of extrapolating entity positions.

A simulation model was also developed for the elliptical trajectory. This showed that at least three times as many updates would be required if the first order algorithm was used in preference to the second order algorithm.

As described in section 2.4, there is ambiguity as to whether a component difference or vectorial difference should be implemented. The current model has the ability to use either method. For a circle, there is little difference between the number of PDUs issued since for large values of radius, only one component changes significantly so that the vectorial difference is very similar to the maximum value of component difference.

For an ellipse, there is more difference between the vectorial and component method of calculating threshold difference than for the circle. Simulations were run varying threshold for an ellipse with major and minor axes 10000 and 6000 m respectively leading to the results in Table 4.

Table 4: PDUs issued for different threshold settings using the elliptical trajectory for both threshold calculation methods – results for the component method are in brackets.

Threshold (m)	PDUs Issued	
	First Order (FPW)	Second Order (FVW)
1	380 (280)	108 (80)
2.5	237 (179)	62 (49)
5	173 (131)	46 (36)
10	122 (91)	35 (28)
20	86 (65)	28 (23)

As expected, the vectorial method yields a greater number of PDUs than the component method since the threshold is exceeded more readily.

3.2 Comparison with Experimental Results

The model was compared with experimental data from a simple DIS-compliant Computer Generated Forces (CGF) model from Mak (www.mak.com). This model uses an aircraft entity that orbits around a fixed point at constant speed. The parameters speed, radius, and dead reckoning algorithm can be readily set while varying the other parameters such as threshold requires editing and recompilation.

Table 5 shows a comparison between the simulation model predictions and the CGF system for several different orbits using the 1 m threshold value.

Table 5: PDUs issued for different radii with threshold set to 1m

Radius (m)	PDU Rate (CGF results in brackets)	
	First Order (FPW)	Second Order (FVW)
1000	3.98 (4.024)	1.05 (1.0526)
2500	2.495 (2.507)	0.586 (0.588)
5000	2.00 (1.818)	0.374 (0.370)
10000	1.333 (1.33)	0.235 (0.235
20000	0.99 (0.959)	0.148 (0.154)

These results demonstrate that the model provides excellent agreement (better than 0.1%) with experimental data providing confidence in its predictive capabilities.

The CGF model was able to use a range of dead reckoning algorithms. Figure 2 shows the times at which the first update PDU is required to be issued for the dead reckoning algorithms supported (1 to 5) and for different values of threshold.

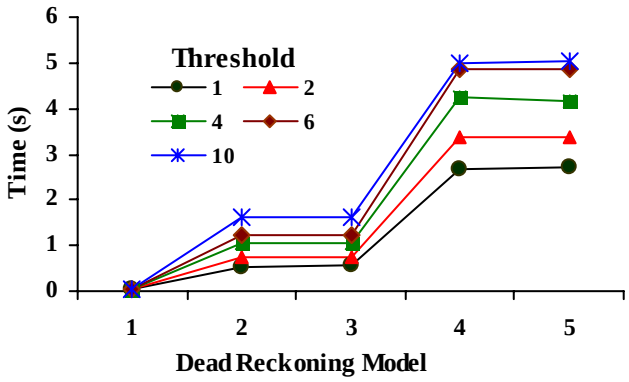

Figure 2: Times at which first dead reckoned PDU is issued for different algorithms and threshold values (each line represents a different threshold setting).

The data clearly show a significant advantage for the second order algorithm (number 5). For each value of threshold it predicts the highest issuance time for an update PDU. Moreover, at the higher threshold settings this time approaches the default heartbeat update time of 5 s. Here, orientation threshold has been set to a high value of 30° so that algorithms 4 and 5 are very similar; if the default value of 3° is used, algorithm 4 (RVW), that includes orientation, is clearly superior.

3.3 Third Order Algorithms

First and second order algorithms have been addressed in the previous sections. In earlier work [9], third order algorithms were also studied. These include the rate of change of acceleration as well as velocity and acceleration to predict future entity positions.

The simulation model was extended to include a sample third order algorithm (from [11]) that used the following formula to predict position:

$$P_1 = P_0 + v_0\Delta t + \tfrac{1}{2}a_0(\Delta t)^2 + \tfrac{1}{6}(\Delta t)^3(a_0 - a_{-1})/\Delta t_{-1}$$

where a_0 is the acceleration at the start of the interval, a_{-1} is the acceleration when the previous update PDU was issued, and Δt_{-1} is the time interval between the previous two update times. This algorithm has at least 10 floating point operations and thus will be more computationally expensive than the first and second order algorithms studied earlier. Note also that this is only one possible third order algorithm; there are 4 other similar algorithms that calculate dead reckoned position using different means of computing change of acceleration [10].

PDU issuance rates and RMS errors were calculated using the model for the third order algorithm for a range of thresholds under the same conditions as in the previous section. These results are shown in Table 6.

Table 6: PDU rates and RMS errors for second and third order algorithms for an aircraft traveling at 200 m/s around a circle of 5000 m radius

Threshold (m)	Second/Third Order PDUs/s	Second/Third Order RMS (m)
0.5	0.464/0.293	0.0037/0.0035
1.0	0.369/0.223	0.0073/0.0073
2.5	0.274/.0.178	0.017/0.017
5	0.216/0.153	0.034/0.033
10	0.172/0.127	0/069/0.066

These results show that the addition of third order terms does not significantly improve the accuracy of the extrapolation so that the additional computation is not warranted. For example, with a 5 m threshold, 0.216 PDUs/s are issued with an RMS error of 0.034 m for the second order algorithm. The third order algorithm predicts a slightly lower PDU issuance rate of 0.153/s but with a similar RMS error of 0.033 m.

3.4 Effect of Heartbeat Mechanism

In the previous sections, the heartbeat rate was set to 1000s so that the threshold dead reckoning mechanism could be studied separately. The default value for heartbeat, however, is 5 s.

Running the model for various values of threshold and radius showed that either the dead reckoning or heartbeat update mechanisms dominate PDU issuance for the circular trajectory. If a high value of threshold is set, all PDUs will be heartbeat updates issued every 5 s. If a low value is set for threshold, all PDUs will be dead-reckoned PDUs.

However, for an elliptical trajectory, there can be a mix of both dead reckoned and heartbeat PDUs since the ellipse has different geometry from the symmetry of the circle. Figure 3 shows a screen shot from the model using an elliptical orbit.

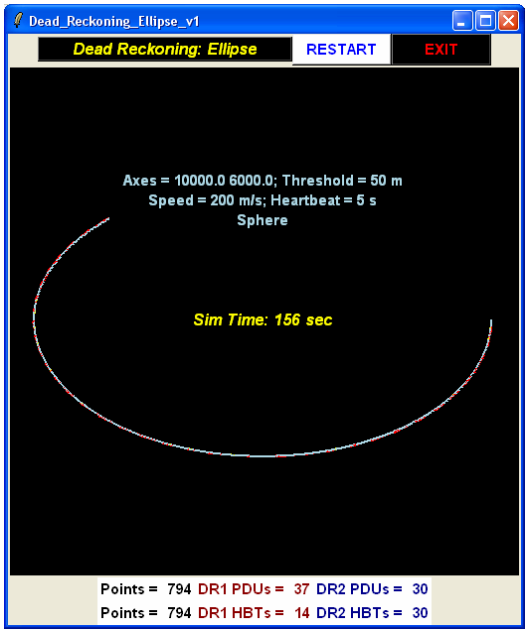

Figure 3: Screen shot from simulation model for elliptical orbit

Here the first order algorithm predicts both heartbeat and dead reckoned PDUs while the second order algorithm predicts only heartbeat PDUs.

3.5 Threshold for Different Entity Types

In many systems, a single threshold value is used for different entity types. However, a fast jet or missile will easily exceed this threshold value in a very short time when manoeuvring at high velocity. It has been suggested (see DIS discussion groups in [4]) that different default values be used for entities operating in different domains such as land, sea and air.

Several methods could be used to better define threshold levels.

1. Specify default threshold levels to be used with different entity types; for example, 1 m for ships; 10 m for aircraft.
2. Alternatively, employ an adaptive thresholding, where the threshold is proportional to the entity's velocity: *p_threshold* = *p_constant* * |*v*|, with v

the given entity's velocity, and |v|>0 and *p_min* <= *p_threshold* <= *p_max*

The simulation model can be used to determine more realistic settings. Varying entity type for different thresholds gives the results shown in Table 7 for various typical speeds of some representative air, land and sea going entities.

Table 7: PDU rates for different threshold values for different entity types travelling at typical speeds for each algorithm in an orbit of 1000 m radius (bolded values indicate that the PDUs are all heartbeat).

Entity Type	**Threshold**	**PDUs/s** FPW/FVW	
Ship at 10 m/s	1 m	0.22	**0.198**
Tank at 20 m/s	2 m	0.31	**0.197**
Maritime Patrol Aircraft at 100 m/s	10 m	0.684	0.238
Fast jet at 250 m/s	25 m	1.074	0.438
Missile at 500 m/s	50 m	1.511	0.716

These results suggest that the relationship between PDU issuance, entity speed, and threshold is highly non-linear so that predicting threshold from a linear relationship as above is inadequate. Increasing speed from 10 m/s to 500 m/s and scaling the threshold linearly from 1 m to 50 m results in a far greater PDU issuance rate (1.511 compared to 0.22) than would be expected from a linear relationship between entity speed and threshold.

4. Conclusions

A simulation model has been developed that predicts PDU issuance rates using both the dead reckoning and heartbeat update mechanisms. The model shows good agreement with experimental results and theory and also shows that second order algorithms result in far lower issuance rates of dead reckoned PDUs.

For a circular orbit, all PDUS issued are either dead reckoned PDUs or heartbeat update PDUs due to symmetry. Other orbits, such as an ellipse, can yield a mix of both PDU types.

This paper has examined positional dead reckoning and heartbeat updates for some simple trajectories. Orientational dead reckoning and the use of more sophisticated manoeuvres will be the subject of a future paper.

5. References

[1] Miller, D.D and J.A. Thorpe, (1995), "SIMNET: The Advent of Simulator Networking", *Proc. IEEE* Vol. 83, No. 8

[2] *DIS Vision: A Map to the Future of Distributed Simulation.* (1994). Prepared by the DIS Steering Committee, Institute of Simulation and Training, University of Central Florida, Orlando, Florida, US

[3] High Level Architecture website on Defense Modeling and Simulation Office (DMSO) website: https://www.dmso.mil/public/transition/hla/

[4] Foundation Initiative 2010 TENA web site: http://www.fi2010.org/index.php

[5] Simulation Interoperability Standards Organisation (SISO) web site (2003): http://www.sisostds.org/

[6] IEEE 1278-1993 (1993), IEEE *Standard for Information Technology - Protocols for Distributed Interactive Simulation Applications*

[7] IEEE 1278.1-1995 (1995), *IEEE Standard for Information Technology - Protocols for Distributed Interactive Simulation Applications*

[8] IEEE 1278.1a-1998, (1998), *IEEE Standards for Information Technology – Protocols for Distributed Interactive Simulation*

[9] Lin, K-C. and D.E. Schab, (1994), "The Performance Assessment of the Dead Reckoning Algorithms in DIS", *Simulation*, Vol. 63, No 5., p 318 - 325

[10] Lin, K-C. and D.E. Schab, (1995), "Network Load in Distributed Interactive Simulation", *J. Aircraft*, Vol. 32, No. 6, p 1392 – 1394

[11] Lin, K_C., Wang, M., and J. Wang, (1996), "Smoothing of Dead Reckoning Image in Distributed Interactive Simulation, *J. Aircraft*, Vol. 33, No.2, p 450 - 452

[12] Durbach, C. and J-M Fourneau, (1998), "Performance Evaluation of a Dead Reckoning Mechanism", Proc. IEEE Second International Workshop on Distributed Interactive Simulation, held Montreal, Canada

[13] Lee, B-S, Cai, W., Turner, S, and L. Chen, (2003), "Adaptive Dead Reckoning Algorithms for Distributed Interactive Simulation", International Journal of Simulation, Systems Science, & Technology, Vo1. 1, 2003 (web reference: http://ducati.doc.ntu.ac.uk/uksim/journal/issue-1/Lee-Turner/B-S%20Lee.pdf)

[14] Aggarwal, S., Banavar, H., Khandelwal, A., Mukherjee, S., and S. Ranjarajan, (2004), "Accuracy in Dead Reckoning based Distributed Multi-Player

Games", Proc. of ACM SIGCOMM 2004, Netgames 2004 Workshop

Author Biographies

PETER RYAN is a Principal Research Scientist in the Defence Science Technology Organisation's Air Operations Division within the Australian Defence Department. He has a background in the modelling and simulation of military operations. His main research interests include Advanced Distributed Simulation, real time simulation, synthetic environments, and their potential to provide enhanced training solutions for the Australian Defence Force. He is also a member of SISO's DIS Product Development Group.

WILL OLIVER holds degrees in Aerospace Engineering and Mathematics. Mr Oliver joined the Air Operations Division of DSTO in 2006 and works in the Advanced Distributed Simulation Laboratory; researching interoperability issues and analysis techniques of advanced distributed simulation. Mr Oliver is a member of the SISO DIS Product Development Group. Prior to joining DSTO Mr Oliver developed software for flight simulators and simulated maintenance trainers.