An Auto-adaptive Dead Reckoning Algorithm for Distributed Interactive Simulation

Wentong Cai Francis B.S. Lee L. Chen
School of Applied Science
Nanyang Technological University
Nanyang Avenue
Singapore 639798
Email: {aswtcai,ebslee}@ntu.edu.sg

Abstract

This paper describes a new, auto-adaptive algorithm for dead reckoning in DIS. In general dead-reckoning algorithms use a fixed threshold to control the extrapolation errors. Since a fixed threshold cannot adequately handle the dynamic relationships between moving entities, a multi-level threshold scheme is proposed. The definition of threshold levels is based on the concepts of area of interest *(AOI) and* sensitive region *(SR), and the levels of threshold are adaptively adjusted based on the relative distance between entities during the simulation. Various experiments were conducted. The results show that the proposed auto-adaptive dead reckoning algorithm can achieve considerable reduction in update packets without sacrificing accuracy in extrapolation.*
Keywords: *Distributed Interactive Simulation (DIS), Dead Reckoning, Relevance Filtering, Extrapolation, Area of Interest (AOI), Sensitive Region (SR), Multi-level Threshold, Adaptive Algorithm.*

1. Introduction

Distributed Interactive Simulation (DIS) is a technology for linking simulations of various types at multiple locations to create a realistic, complex, "virtual world" for the simulation of highly interactive activities. The High Level Architecture (HLA) [10] is a general purpose architecture designed to promote simulation reuse and interoperability. This is achieved through the HLA concept of the federation: a composable set of interacting simulations. Moreover, the DIS community has developed the real-time platform reference federation object model (RPR FOM) [9] to provide the functionality of the DIS standard within the HLA environment.

Since simulation entities are physically distributed in DIS, for a large scale DIS exercise, updating states of the simulation entities may generate a large amount of communication and thus saturate network bandwidth. To reduce the amount of communication, *dead reckoning* (DR) technique, a fundamental feature of the DIS standard [5], was developed.

Previous research works on dead reckoning have been largely focused on the evaluation of extrapolation equations [4, 6], and the performance investigation of DR mechanisms (e.g., [2, 3]). In this paper, we introduce an adaptive dead reckoning algorithm that reduces the number of state update packets without sacrificing extrapolation accuracy. The performance of the algorithm will also be studied.

In general, dead reckoning algorithms use a fixed threshold, regardless of the relationship between the entities, to control errors in extrapolation. In order to maintain an adequate accuracy, a small threshold is usually used. However, since the exact position is not important for entities that are far away from each other, unnecessary update packets may be generated. To reduce the number of update packets while maintaining adequate accuracy, in our adaptive dead reckoning algorithm, the threshold is dynamically changed according to the relative distances between simulation entities. Concepts found in *relevance filtering* are used in our definition of threshold levels.

Our work is different from the works done in the areas of relevance filtering (e.g., [1, 7, 8]), although we have the same goal, that is, to reduce the traffic on the network and to improve the scalability of DIS systems. Relevance filtering is concerned with eliminating the transmission of irrelevant packets, whereas, our work focuses on reducing the number of state update packets

	One-Step	*Two-Step*
1^{st} Order	$x_t = x_{t'} + v_{t'}\tau$	$x_t = x_{t'} + \frac{x_{t'} - x_{t''}}{t' - t''}\tau$
2^{nd} Order	$x_t = x_{t'} + v_{t'}\tau + 0.5a_{t'}\tau^2$	$x_t = x_{t'} + v_{t'}\tau + 0.5\frac{v_{t'} - v_{t''}}{t' - t''}\tau^2$

Table 1. Extrapolation Equations

caused by the dead reckoning mechanism.

This paper is arranged as follows: Section 2 will give a brief introduction on dead reckoning. The definition of threshold levels and our adaptive dead reckoning algorithm will be covered in Section 3. Section 4 will report some preliminary experiment results, and Section 5 will conclude the paper and outline our future works.

2. Dead Reckoning

One of the important aspects in DIS is the ability of each simulator to represent accurately in real-time the state of all simulation entities, including both local and remote, participating in the same DIS exercise. To reduce the number of state update packets, the DR technique is used. In addition to the *high fidelity model* that maintains the accurate position about its own simulation entities, each simulator also has a *dead reckoning model* that estimates the position of all simulation entities (both local and remote). The anticipated position of an entity is usually calculated based on the last (or past) accurate state information of the entity using an *extrapolation equation* (see Table 1). So, instead of transmitting state update packets, the estimated position of a remote simulation entity can be readily available through a simple, local computation.

To maintain the accuracy, after each update of its own simulation entity, a simulator needs to compare the true position of the entity obtained from the high fidelity model and its extrapolated position. If the difference between the true and the extrapolated position is greater than a pre-defined *threshold*, a state update packet will need to be sent to other simulators. Extrapolation for the entity will then be corrected by all dead reckoning models at all simulators, based on the updated state of the entity.

Threshold is an important parameter in the DR algorithm. It is used to control the accuracy of extrapolation, and affects the number of entity state update packets generated. A small threshold makes DR algorithm generate state update packets at a higher frequency, but results in higher accuracy in the estimation of the entity's position. On the other hand, the DR algorithm using large threshold generates fewer update packets, but its accuracy is also lower.

Current extrapolation equations can be divided into two groups: one-step formulas and multi-step formulas [6]. One-step formulas only use the last state update packet to extrapolate an entity's position, whereas, multi-step formulas use the last two or more state update packets in the extrapolation. In our experiments, one-step, second order formula is used.

In Table 1, $x_{t'}, v_{t'}$ and $a_{t'}$ represent respectively the position, velocity and acceleration of the entity as found in the last state update packet. Similarly, $x_{t''}, v_{t''}$ and $a_{t''}$ are the position, velocity and acceleration of the second last state update packet. τ is the elapse time from the last update. The formulas are used to extrapolate the position of the entity at time $t = t' + \tau$.

3. Adaptive Multi-level Threshold

Generally, DR algorithms are implemented with a fixed threshold. Although it is easy for the simulator to operate the DR model, a fixed threshold may not adequately handle the dynamic relationship between entities. In fact, the value of the threshold is relevant to the distance between entities. In DIS, each entity, like the real object it simulates, may have different interests in other entities around it. For example, when two entities are close to each other, each of them may need to pay more attention to the other's movement. In this case, a small threshold should be used because the accuracy of extrapolation is important. However, when two entities are far away from each other, the position of one entity may become irrelevant to the other entity. Thus, a large threshold will be sufficient under this situation.

To determine the levels of threshold, concepts in *relevance filtering* [7] are used. In relevance filtering, *area of interest* (AOI) of an entity, also called *reachability region* in [1], is defined as a circle with a constant radius around the entity. The length of radius is usually defined according to the entity type. Hence, the AOI of all entities (either local or remote) can be easily determined by a simulator. In addition to AOI, an entity also has its *sensitive region* (SR). If one entity moves into another entity's SR, a collision will likely happen.

Level	4	3	2	1
Description	no overlap of AOIs	overlap of AOIs with another entity	in another entity's AOI	in another entity's SR

Table 2. Threshold Levels

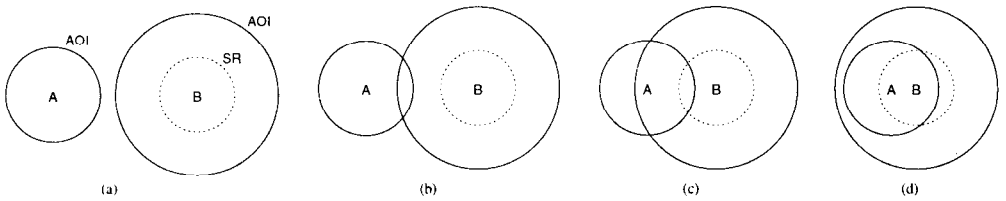

Figure 1. Multi-level Threshold Illustration

By using AOI, SR and the distance between entities, four threshold levels can be defined, with level 1 being the smallest threshold and level 4 the largest. They are listed in Table 2 and illustrated in Figure 1. In the figure, the circle with a solid line represents the entity's AOI, and the circle with a dotted line represents the entity's SR.

When an entity A's AOI is not overlapped with any other entity's AOI (part (a) of Figure 1), the level 4 threshold will be used in entity A's extrapolation. Large errors are allowed in the extrapolation, and update packets will be sent at a low frequency. In this case, entity A's movement is not interesting to other entities, and extrapolation can be less accurate.

When the entity A's AOI is overlapped with the entity B's AOI, the extrapolation of entity A's movement needs to be more accurate. There are three cases:

- If entity A is in entity B's SR (part (d) of Figure 1), to prevent entity B from making misjudgment on collision, level 1 threshold (that is, the smallest threshold) has to be used in entity A's extrapolation. In this case, entity A's update packet will be emitted most frequently, but its extrapolation will be the most accurate.

- If entity A is outside entity B's SR but within its AOI (part (c) of Figure 1), level 2 threshold will be adopted. To avoid missing the detection of an approaching entity, entity B needs to have a more accurate position of entity A. However, there is no danger of making misjudgment on collision. Therefore, a small threshold will be adequate in entity A's extrapolation.

- If entity A is outside entity B's AOI (part (b) of Figure 1), level 3 threshold will be used. Since the two AOIs are overlapped, one entity may move to another's AOI in a short period of time. The extrapolation of A's position still needs to be accurate, but the requirement is less rigid. Entity A's update packets do not need to be sent frequently because A is not in B's AOI. So, a relatively large threshold can be used in entity A's extrapolation.

In all these cases, extrapolation of A needs to be accurate, though the degree of accuracy required is different.

Threshold values at different threshold levels may be determined by factors in the simulation environment or entity characteristics. For example, the value of level 4 threshold could be limited by an entity's AOI, and the value of level 1 threshold may be defined by the *collision distance* between entities. In a DIS exercise, any entity's movement is restricted by the network delay and the time spent by the simulator to process simulation events. For this reason, judgment of collision of simulation entities is not like that of real objects. Generally, if the distance between two entities is less than a pre-defined collision distance, a collision is considered to have happened in the simulation.

For each local simulation entity `e` that belongs to a simulator i, these four levels of threshold will be used in the extrapolation. If an error is detected at a certain threshold level k, an update packet will be sent only to the simulators with a remote entity `re` whose threshold level is k or less. An update packet will be sent to all other simulators only when an error is detected at the threshold level 4. In this way, update packets will be sent only to the relevant entities, and the amount of communication will be reduced.

For a local entity `e` and a remote entity `re`, `TLevel[e,re]` gives the threshold level used to decide when to send `re` `e`'s update packet. It determines how often `re` may receive `e`'s update packets. Different re-

```
/* for simulator i */
/* assume that ∀ e and re, TLevel[e,re] = 4 initially */
for every received packet of remote entity re do
    for each local entity e ∈ simulator i do {
        Dist = distance(e,re);
        if Dist < AOI(e) + AOI(re) then
            if Dist < SR(re) then
                TLevel[e,re] = 1;
            else
                if Dist < AOI(re) then
                    TLevel[e,re] = 2;
                else
                    TLevel[e,re] = 3;
        else
            TLevel[e,re] = 4;
    }
```

Figure 3. Algorithm to Determine Threshold Level

```
/* for simulator i */
for each state update of local entity e do {
    extrapolate e's position based on the past state information;
    /* send update packet if necessary */
    switch Diff = |TruePosition - ExtrapolatedPosition | {
        case Diff > LevelOneThreshold
            multicast an update packet to the group
                {simulator k | re ∈ simulator k ∧ TLevel[e,re] = 1};
            break;
        case Diff > LevelTwoThreshold
            multicast an update packet to the group
                {simulator k | re ∈ simulator k ∧ TLevel[e,re] ≤ 2};
            break;
        case Diff > LevelThreeThreshold
            multicast an update packet to the group
                {simulator k | re ∈ simulator k ∧ TLevel[e,re] ≤ 3};
            break;
        case Diff > LevelFourThreshold
            broadcast an update packet to all other simulators;
            break;
        default
            break;
    }
}
```

Figure 4. Algorithm to Send Update Packets

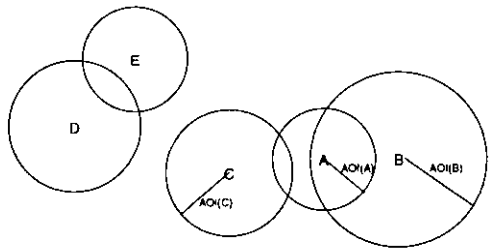

Figure 2. Multi-level Threshold Example

mote entities may have different threshold levels. For example, in Figure 2, different threshold levels will be used in entity A's extrapolation. Update packets will be sent only to entity B when an extrapolation error happens at threshold level 2. If an extrapolation error occurs at threshold level 3, an update packet will then be sent to both entities B and C. An update packet will be sent to all other entities (that is, entities B, C, D and E) when an error happens at threshold level 4.

Our adaptive, multi-level dead reckoning algorithm, therefore, consists of two parts:

- For each remote entity re, to determine its threshold level when its update packet is received; and
- For each local entity e, to determine the sending of its update packet when its state changes.

The first part of the algorithm is given in Figure 3, where function $AOI(\mathtt{e})$ returns the radius of AOI of entity e, and function $SR(\mathtt{e})$ returns the radius of SR. The second part of the algorithm is given in Figure 4.

4. Experiment Results

To test the algorithm, two experiments were conducted. In both experiments, a simulated environment was created and a program was written to simulate what should happen in a distributed environment. Statistics, such as total number of update packets generated and average extrapolation accuracy, were collected during the simulation.

4.1. Experiment One

In the first experiment, there are two entities: one is a *motionless entity* (ME) at position (110, 0); the other entity (referred to as a *test entity*, TE) is moving along a circle with center (0, 0) and radius 100. The radiuses of ME's SR and AOI are 10 and 50 respectively. The level 1, 2, 3 and 4 threshold values of TE are 5, 10, 20 and 40 respectively. The collision distance is assumed to be 5. The unit used for distance in the experiment is meter.

Figure 5 shows the the trajectory of TE's movement (that is, the smooth sinusoid curve) and its extrapolation in x dimension when the fixed threshold values are used. Note that the trajectory of ME is the straight line at 110. The number of update packets (that is, DRNUM) is also shown in the figure. When the threshold is 5 (Figure 5(a)), the extrapolation is sufficiently accurate for ME to make correct collision decisions. When the threshold is 10 (Figure 5(b)), since the extrapolation is less accurate, the ME may make some wrong collision judgments as TE is moving close to it. However, fewer update packets are generated in this case (14 compared to 19).

Figure 6 shows the trajectory of TE's movement when the threshold value is adjusted adaptively. It can be seen that when TE is approaching ME, the extrapolation error becomes smaller and smaller. Therefore, the extrapolation of TE at the closest point to ME is accurate enough for ME to make the correct judgment. In addition, since a large threshold will be used when TE is far away from ME, the number of update packets generated is also low. In fact, the least number of update packets (that is, 13) is generated in this case. So, by adaptively adjusting threshold values, we can achieve the required accuracy as in the case shown in Figure 5(a) as well as the low generation rate of update packets as in the case shown in Figure 5(b).

4.2. Experiment Two

In the second experiment, an entity may move randomly in a $500m \times 700m$ two dimensional space, and is represented by a circle with radius of $2.5m$. For this simple experiment, we assume that all entities have the same AOI and SR. The radiuses of an entity's AOI and SR are defined as $32m$ and $10m$ respectively. The number of entities in the experiment varies from 16 to 32, and each entity is simulated by a simulator. The position of each entity is updated every 5 *sec*. At each update, the speed and direction of each entity may change randomly. The maximum speed is 3 m/sec, and the maximum acceleration is 0.5. The simulation duration is 10 minutes.

In our experiments, the extrapolation accuracy is defined by *average error in AOI* and *average error in SR*. The average error in AOI is calculated using the procedure shown in Figure 7. For each position update, the average error between the real and the extrapolated positions is calculated. For each entity e, we are only interested in those entities whose real position is in e's AOI (that is, the set $S_{\mathtt{e}}$). The average error in SR can

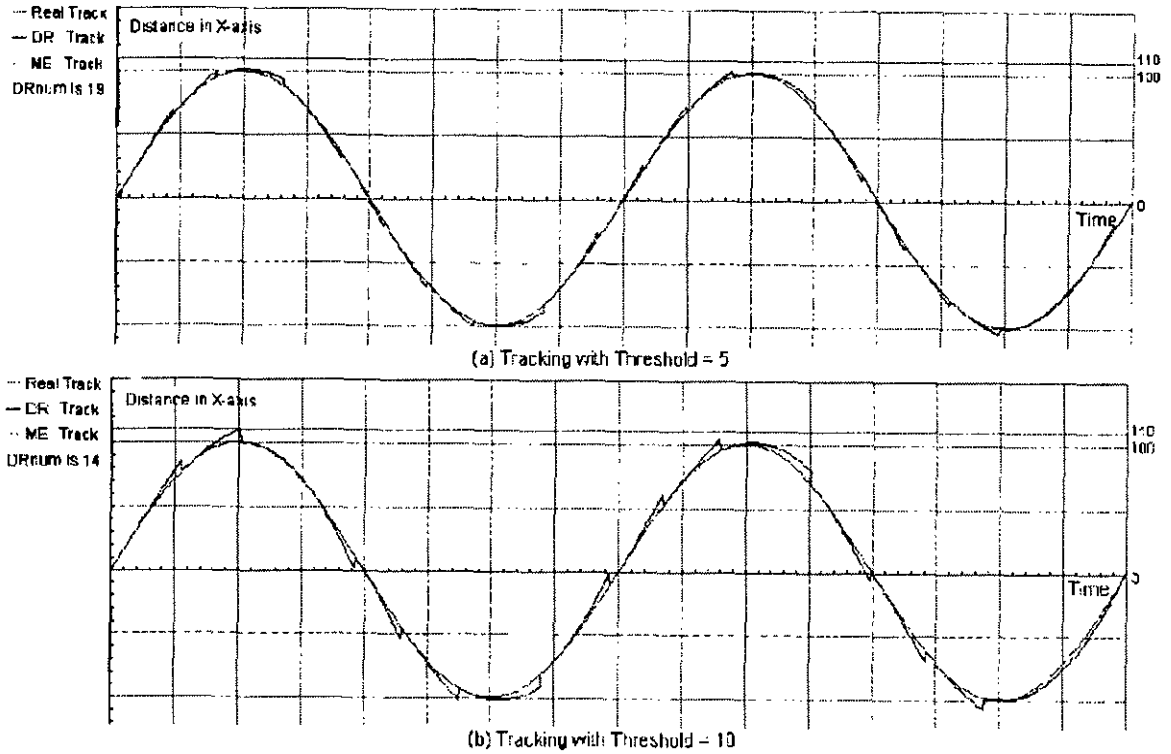

Figure 5. Extrapolation with Constant Threshold

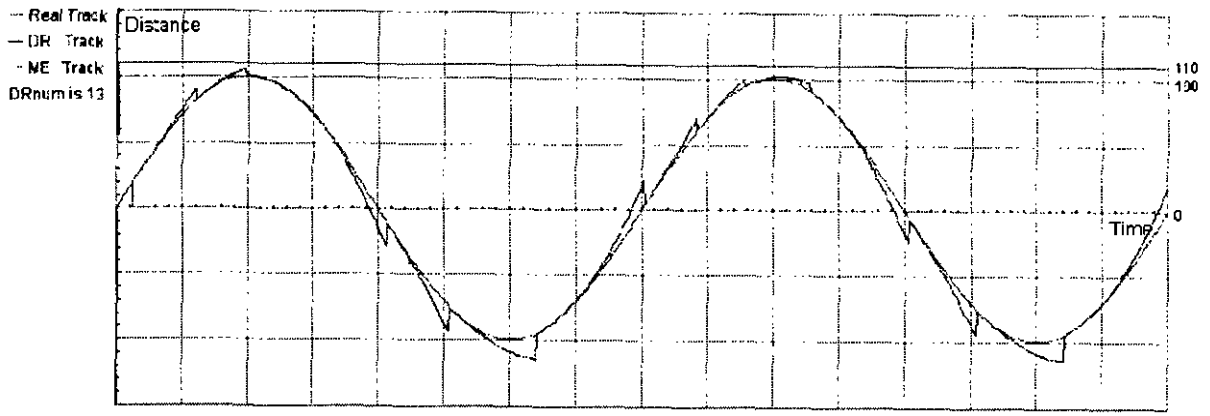

Figure 6. Extrapolation with Multi-level Threshold

be calculated similarly if SR is used instead of AOI in S_e's definition.

threshold	2	8	12	18	24
# of PDUs	4844	3380	3079	2779	2624
Avg E in AOI	0.5608	2.6838	3.5187	5.5289	7.5277
Avg E in SR	0.5457	2.1462	3.0143	5.4567	7.1402

16 entities in the simulation

Threshold	2	8	12	18	24
# of PDUs	9705	6796	6203	5638	5230
Avg E in AOI	0.5645	2.4171	3.0295	5.6765	8.6891
Avg E in SR	0.5573	2.4637	3.6056	5.1140	6.7460

32 entities in the simulation

Table 3. Fixed Threshold Results

Table 3 shows the number of PDU (Packet Data Unit) generated and the average error in AOI and SR when fixed threshold is used in the dead reckoning algorithm. (The unit of threshold is meter.) The following are some observations obtained from these numbers:

- As the threshold used increases, fewer PDUs will be generated, but the average error in AOI (and SR) increases.
- In most of the cases, there is no large difference between the average error in AOI and the average error in SR, since the fixed threshold is used.
- Except when the smallest threshold (i.e., 2) is used, for all other cases the average errors are large, compared to the entity's size.

# of Entities	16	32
# of PDUs	3461	7875
Avg Error in AOI	1.0130	1.4012
Avg Error in SR	0.4990	0.5414

Table 4. Multi-Level Threshold Results

Threshold Level	4	3	2	1
Value (in Meters)	30	18	8	2

Table 5. Multi-Level Threshold Values

Table 4 shows the number of PDUs generated and the average error in AOI and SR when our multi-level threshold dead reckoning algorithm is used. The threshold values used in different levels are listed in Table 5.

It can be seen from Table 4 that there is a great reduction in the average error in SR, compared to the average error in AOI. In our algorithm, if entity A is in entity B's SR, a minimum threshold will be used in the dead reckoning so that B will receive A's update packets most frequently. In this case, B will know A's position most accurately.

Compared to the case where the threshold is fixed at the minimum (that is, 2), using multi-level threshold, the reductions on the number of PDUs are 29% and 19% for 16 and 32 entities respectively. The average errors in AOI increase. But, the average errors in SR are unaffected in both scenarios. Compared to the case where the threshold is fixed at 8, the number of PDUs increases slightly when multi-level threshold is used. The percentage increments are 2% and 14% respectively for 16 and 32 entities. However, the average errors in SR are reduced greatly by about 77% for both situations. The average errors in AOI are also reduced.

In summary, the above two experiments demonstrate the feasibility of our adaptive, multi-level threshold dead reckoning algorithm. The results show that using multi-level threshold in dead reckoning, adequate extrapolation accuracy can be achieved with the reduced number of state update packets.

5. Conclusions

This paper describes a new, auto-adaptive algorithm for dead reckoning in DIS. Since using a fixed threshold to control extrapolation error may either generate unnecessary update packets (when threshold is small) or result in mistakes in the simulation (when threshold is large), a multi-level threshold scheme is proposed in the paper. The definition of threshold levels are based on the concepts of AOI and SR, and the levels of threshold are adaptively adjusted based on the relative distance between entities during the simulation. Depending on the threshold level, an update package is sent only to the relevant entities. Thus, a close-by entity may receive more update packages than an entity that is far away. To test the scheme, two experiments were conducted. The results show that the multi-level thresholds can adequately reflect the dynamic relationship between entities. It reduces the rate of transmitting update packages while maintaining adequate accuracy in the extrapolation.

In our future work, we will look at the implementation details of the algorithm. In particular, we will study how this algorithm can fit into the framework

```
SumOfError1 = 0;
NumUpdates = 0;
/* (a) calculate the sum of average errors for all updates */
for each position update do {
    NumEntities = 0;
    SumOfError2 = 0;
    /* (b) calculate the sum of average errors for all entities in each update */
    for each entity e do {
        SumOfError3 = 0;
        /* (c) calculate the sum of errors relative to an entity e */
        Se = {e' | e' real position in e's AOI};
        for each entity e' in Se do
            SumOfError3 = SumofError3 + the difference between
                the real position of e' and its extrapolated position;
        SumOfError2 = SumOfError2 + SumOfError3 / SizeOf(Se)
        NumEntities++;
    }
    SumOfError1 = SumOfError1 + SumOfError2 / NumEntities;
    NumUpdates++;
}
/* (d) calculate the average error in AOI */
AvgErrorInAOI = SumOfError1 / NumUpdates;
```

Figure 7. Procedure to Calculate The Average Error in AOI

of the High-Level Architecture (HLA). We will also investigate a more systematic approach to defining the threshold values for different threshold levels. In addition, further experiments will also be conducted for more realistic scenarios.

References

[1] M. Bassiouni, M.-H. Chiu, M. Loper and M. Garnsey. Relevance Filtering for Distributed Interactive Simulation. *Computer Systems - Science & Engineering*, Vol.13, No.1, pp.39-47, Jan 1998.

[2] Corentin Durbach and Jean-Michel Fourneau. Performance Evaluation of a Dead Reckoning Mechnism. In Proc. of IEEE 2nd Int. Workshop on Distributed Interactive Simulation and Real-time Applications, pp.23-29, July 1998.

[3] G. Figart, C. Slaton and S. Slosser. Using Encumbrance Factors to Improve Dead Reckoning Approaches to Object Regeneration and Modeling. In Proc. of 14th DIS Workshop on Standards for the Interoperability of Distributed Simulation, March 1996.

[4] L. Foster and P. Maassel. The Characterization of Entity State Error and Update Rate for DIS. In Proc. of 11th DIS Workshop on Standards for the Interoperability of Distributed Simulation, pp.61-73, Sept. 1994.

[5] IEEE 1278. Standard for Information Technology - Protocols for Distributed Interactive Simulation Applications. 1993.

[6] Kuo-Chi Lin. Dead Reckoning and Distributed Interactive Simulation. In Proc. of SPIE Conference (AeroSense'95), Orlando Florida, April 1995.

[7] Katherine L. Morse. Interest Management in Large-Scale Distributed Simulation. Dept. of Information & Computer Science, Univ. of California at Irvine, Technical Report #96-27, 1996.

[8] S.J. Rak and Daniel J. van Hook. Evaulation of Grid-based Relevance Filtering for Multicast Group Assignment. In Proc. of 14th DIS Workshop on Standards for the Interoperability of Distributed Simulation, pp.739-747, March 1996.

[9] G. C. Shanks. The RPR FOM, A Reference Federation Object Model to Promote Simulation Interoperability. In Proc. Spring Simulation Interoperability Workshop, 1997.

[10] U.S. Department of Defense. *High Level Architecture Interface Specification*, Version 1.3, April 1998. (Available at: http://www.dmso.mil/hla/)