

# Scalable Interest Management for Multidimensional Routing Space

Elvis S. Liu  
mcelvis@polyu.edu.hk

Milo K. Yip  
mcmilo@polyu.edu.hk

Gino Yu  
mcgino@polyu.edu.hk

Multimedia Innovation Centre  
School of Design  
The Hong Kong Polytechnic University

## ABSTRACT

Interest management is essential for scalable collaborative virtual environments (CVEs) which sought to reduce bandwidth consumption on the network. Most of the interest management systems such as Data Distribution Management (DDM) service of the High Level Architecture (HLA) concentrate on providing precise message filtering mechanisms. However, in doing so a second problem is introduced: the CPU cycle overheads of filtering process. If the cost in terms of computational resources of interest management itself is too high, it would be unsuitable for real time applications such as multiplayer online games (MOGs) for which runtime performance is important. In this paper we present a scalable interest management algorithm which is suitable for HLA DDM. Our approach employs the collision detection method of I-COLLIDE for fast interest matching. Furthermore, the algorithm has been implemented in our commercialized MOG middleware - Lucid Platform. Experimental evidence demonstrates that it works well in practice.

## Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed System—*Client/Server, Distributed Applications*; I.6.7 [Simulation and Modeling]: Simulation Support Systems

## Keywords

Computer Games, Collaborative Virtual Environments, Interest Management, Collision Detection, High Level Architecture, Data Distribution Management

## 1. INTRODUCTION

Collaborative virtual environments (CVEs) allow multiple users to interact in real-time even though those users are located differently around the world. Since the success of military simulation CVEs such as SIMNET [5] and DIS

[1] in early 1990s, the attention of CVE development was no longer restricted to the research communities. Commercial applications such as multiplayer online games (MOGs) were also available for entertainment purpose. Recently, high bandwidth and low latency network environments allows the gaming industry to develop massively multiplayer online games (MMOGs) which support thousands of geographically distributed players to participate in the same shared virtual world. As the CVE grows in terms of users and simulation entities, using scalable data replication schemes becomes one of the major requirements for CVE development. Since entity state broadcasting consumes significant network resources, message filtering by interest management schemes was introduced to address the problem.

The basic idea of interest management is very simple. All participants should only receive data of the simulation entities they interest, rather than the entities in the whole shared virtual world. Various data filtering mechanisms have been proposed based on geographical distribution of the simulation entities, which are mainly divided into two categories: zone-based [11] and aura-based [9]. These approaches, however, do not concern filtering aspects other than spatial information. For example, in a military simulation, a low rank soldier may not receive information that sends to his nearby generals. This can be done by additional filtering mechanisms provided by the High-Level Architecture (HLA) [3, 14] such as class-based filtering and value-based filtering through multidimensional routing space.

The HLA message filtering mechanisms are very precise which ensure the participants receive the minimal set of entity states they interest. However, the more complex the filtering mechanisms, the more computational resources are required to process interest matching between subscribers and simulation entities. Consider a CVE with  $n$  update regions associated with the objects and  $m$  subscription regions specified by the participants, using brute force approach to match their interest is an  $O(nm)$  process. The situation becomes worse when the concept of multidimensional routing space of HLA is introduced, the complexity of matching process in terms of number of comparisons becomes  $O(nmd)$  where  $d$  is the number of dimensions of the routing space. Additional process may also be required to match the update and subscription classes as well as the attribute sets for the class-based filtering mechanism. For real-time applications such as MMOGs, the simulation entities may modify their update regions at every frame; clearly the brute force

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

VRST'05, November 7–9, 2005, Monterey, California, USA.  
Copyright 2005 ACM 1-58113-098-1/05/0011 ...\$5.00.

matching process is time consuming and is not scalable for large-scale CVEs.

In this paper we present a scalable algorithm for multidimensional routing space which speeds up the pair-wise interest matching process by caching the results of previous frames. Our approach employs the Sweep-and-Prune algorithm of the classic collision detection system I-COLLIDE [6], by doing so we are able to successfully reduce the average runtime complexity from  $O(nm)$  to  $O(n + m)$ . We demonstrate by experimental evidence that our approach works well in practice. Moreover, the algorithm has been successfully applied to Lucid Platform 1.0 [4] which is an academic developed commercial MOG middleware.

## 2. RELATED WORKS

In this section we briefly review the data distribution management service of the HLA and the existing approaches of collision detection algorithms.

### 2.1 Overview of High-Level Architecture

The HLA is developed by the Defense Modeling and Simulation Office (DMSO) of U. S. Department of Defense which subsequently became IEEE standard of computer simulation (IEEE 1516) [13] in 2000. The HLA specification provides a general infrastructure and services for distributed simulations which consists of three core components including federation rules, interface specification and the object model template (OMT). A Run Time Infrastructure (RTI) is a software implementation that meets the HLA interface specification and executes the federation execution (the simulation system). Six services, namely: federation management, object management, declaration management, ownership management, time management and data distribution management, are provided by the RTI. Our research mainly focuses on message filtering within the data distribution management (DDM) service.

The DDM provides flexible mechanism for publishing and subscribing interests through multidimensional routing space. The basic structure of routing space is as following:

*Routing space:* A routing space is a collection of dimensions.

*Dimension:* Dimensions are used to define regions.

*Extent:* An extent is a bounded range defined along each dimension of a routing space.

*Region:* Each region is defined in terms of a set of extents.

The interest matching process begins by specifying subscription and update regions. An object is said to be interested by a federate if and only if at least one of the object's attributes is subscribed by the federate (through declaration management) and at least one update region associating with the object overlaps the subscription region(s) of the federate. In DDM, the concept of *region* is very flexible: it can be used to define static zones for zone-based filtering, some early research on static multicast grouping [2] are based on this idea. In addition, *region* can also be used for dynamic multicast grouping [12] and aura-based interest management. For the latter case, the subscription and update regions may need to be modified as the states of objects change over time. Such modifications may cause the  $O(nm)$  matching problem and place high demands on computational resources especially for remapping dynamic multicast groups. One approach to dealing with this problem is to use infrequent region update with *lookahead* time

[7] prediction. Specifically, the regions are only modified after a period of time (not at every time step) and they must be sufficiently large so that the simulation entity would not move out of scope before they are updated. This lowers the demand of remapping operations, however, federates may receive irrelevant data with larger subscription regions.

### 2.2 Collision Detection Algorithms

Real-time collision detection has been extensively studied in many fields such as computer graphics, robotics, simulation of mechanics systems and virtual environments for more than a decade. Originally, using the brute force approach to detect collision for a scene of  $n$  objects is an  $O(n^2)$  process (since there are  $C_2^n$  potentially collided object pairs), which is very similar to the object/subscriber matching process of interest management. Throughout the years, many approaches were developed to reduce the  $O(n^2)$  computational complexity. Spatial decomposition techniques [16, 17, 15] are the most common strategies and especially efficient for static scenes. The idea behind these approaches is to decompose the space the objects occupy, so that we only need to detect collision between those pairs that are in the same or nearby clusters of the decomposition, thus to reduce the number of pair-wise comparisons. Hierarchies of bounding volumes [10, 8] have also been a very popular technique for collision detection process. The basic idea is to approximate the objects with bounding volumes such as spheres, axis-aligned bounding boxes (AABBs), oriented bounding boxes (OBBs) and discrete orientation polytopes (k-dops) and represent them in hierarchies which allows for efficient collision queries.

#### 2.2.1 Sweep-and-Prune Algorithm

The Sweep-and-Prune algorithm employed by I-COLLIDE [6] is one of the fastest collision detection methods which involves using AABBs in conjunction with coordinate sorting. The following two ideas are the basis of the Sweep-and-Prune algorithm:

*Dimension Reduction:* If two bounding boxes overlap in three-dimensional space, then all their orthogonal projections on the x-, y-, and z- axes must overlap.

*Temporal Coherence:* Assuming the time steps are small enough that the objects does not travel large distances between two consecutive simulation steps.

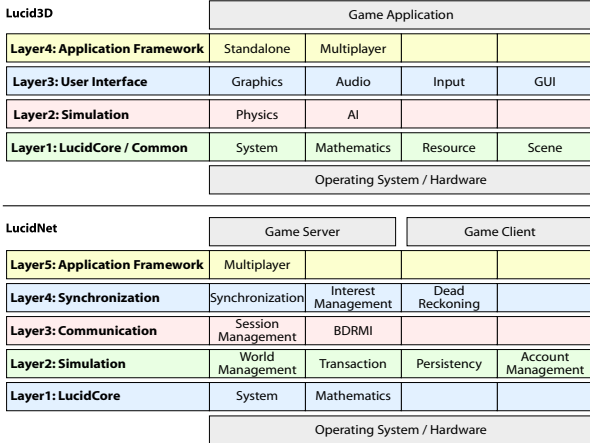
The Sweep-and-Prune algorithm begins by projecting each AABB surrounding an object onto the x-, y-, and z- axes. Three lists will be constructed, one for each dimension. Each list contains the values of the endpoints of the intervals in each corresponding dimension. By sorting these lists, we can determine which intervals overlap as well as which AABBs are collided. In general case, such a sort would take  $O(n \log n)$  time. We can reduce the complexity by keeping the sorted lists from the previous time step. Since Temporal Coherence is exploited, the lists will be nearly sorted, re-sort them using insertion sort is in expected linear time  $O(n)$ .

## 3. INTEREST MANAGEMENT OF LUCID PLATFORM

The Lucid Platform [4] is a middleware for computer game development which has been commercialized to provide complete solution to both single player games and multiplayer online games. The middleware consists of two software development kits, namely, Lucid3D and LucidNet. Lucid3D

aims to provide front-end functionalities such as 3D graphics, audio and AI, where LucidNet makes use of client/server communication model to provide backend functionalities.

**Figure 1** shows the architecture of Lucid Platform 1.0.



**Figure 1: Architecture of Lucid Platform 1.0**

The interest management module within LucidNet provides class-based/attribute-level filtering mechanism and value-based filtering mechanism through multidimensional routing space. Although we did not implement the module strictly according to HLA Interface Specification [3], it was built based on the concepts of declaration management and data distribution management services. Since runtime performance is an important factor for real-time applications, we employed an efficient collision detection algorithm for the interest matching process.

### 3.1 Dimension Reduction

We aim to develop a scalable matching algorithm for multidimensional routing space. Although the spatial decompositions and bounding volume hierarchy approaches are common for fast object culling, they are not suitable to our system because they concern only the 3D space. The Sweep-and-Prune algorithm, however, does not have this limitation. Its dimension reduction concept can be expanded to allow multidimensional overlapping test:

*Two regions overlap in  $n$ -dimensional space if and only if their extents on the 1<sup>st</sup>, 2<sup>nd</sup>, ..., and  $n$ <sup>th</sup> dimension overlap.*

Note that a *region* in Lucid Platform can be defined as a subscription region, an update region, or both.

### 3.2 Class-based Interest Matching

In our implementation, a set of class-based subscriptions  $\mathcal{S}$  and a set of object updates  $\mathcal{U}$  are associated with each region of routing space. When a subscription region and an update region overlap, further investigation of  $\mathcal{S}$  and  $\mathcal{U}$  of the corresponding regions is necessary. Since Lucid Platform allows dynamic change of class-based subscriptions and publications, federates need not to specify their interests statically before joining the execution. The class-based subscription and object update are defined as following:

Class-based subscription:  $S=(F,C_S,A_S)\in\mathcal{S}$ , where  $F$  is the federate,  $C_S$  is the object class and  $A_S$  is the attribute handle set.

Object update:  $U=(O,C_U,A_U)\in\mathcal{U}$ , where  $O$  is the object

instance,  $C_U$  is the object class and  $A_U$  is the attribute handle set.

$O$  is said to be interested by  $F$  if and only if  $C_S=C_U$  and  $A_S\cap A_U\neq\emptyset$ . When  $O$  is updated, the values of the result of  $A_S\cap A_U$  will be sent to  $F$ .

### 3.3 Sorting and Matching

We now discuss our matching algorithm in more detail. Similar to the Sweep-and-Prune algorithm, we first construct a list of endpoints of the extents for each dimension. By sorting these lists (using insertion sort), we can determine which extents overlap. If the extents of two regions,  $R_1$  and  $R_2$ , overlap in all dimensions, the following algorithm for class-based interest matching would be carried out.

```

 $\mathcal{S}_1$ : A set of class-based subscriptions associated with  $R_1$ 
 $\mathcal{U}_1$ : A set of object updates associated with  $R_1$ 
 $\mathcal{S}_2$ : A set of class-based subscriptions associated with  $R_2$ 
 $\mathcal{U}_2$ : A set of object updates associated with  $R_2$ 

BEGIN
  FOR each subscription  $S=(F,C_S,A_S)\in\mathcal{S}_1$ 
    FOR each object update  $U=(O,C_U,A_U)\in\mathcal{U}_2$ 
      IF  $C_S=C_U$  and  $A_S\cap A_U\neq\emptyset$ 
        Establish connection between  $O$  and  $F$ 
      END IF
    END FOR
  END FOR
  FOR each subscription  $S=(F,C_S,A_S)\in\mathcal{S}_2$ 
    FOR each object update  $U=(O,C_U,A_U)\in\mathcal{U}_1$ 
      IF  $C_S=C_U$  and  $A_S\cap A_U\neq\emptyset$ 
        Establish connection between  $O$  and  $F$ 
      END IF
    END FOR
  END FOR
END

```

The overlap status and the class-based matching results are only modified when the insertion sort performs a *swap*. Since temporal coherence is exploited, we can cache the matching results of the previous time steps. In environments where the objects make relatively small movements between consecutive time steps, the lists would remain almost sorted. In general, the complexity of this sorting process is  $O(n+m)$  for each dimension, where  $n$  is the number of the update regions and  $m$  is the number of subscription regions.

### 3.4 Multicast Grouping

One thing we need to point out is that the connection pattern of multicast grouping is independent of our interest matching algorithm. In the implementation of Lucid Platform 1.0, we employed group-per-object multicasting allocation. Therefore, federates with the same interest in an object instance form a multicast group and the updates of the object only transmit to the group members. This approach ensures federates receive the optimal set of data. Alternatively, other connection patterns can also apply to our system without violation of the principles of our algorithm.

## 4. EXPERIMENTAL RESULTS

In this session we describe the evaluation of our proposed interest management algorithm. Several simulation experiments, based on Lucid Platform 1.0, were carried out to

compare the performance between our approach and the brute force approach. The tests were run on a Pentium 4 2.8GHz with 1GB main memory. The followings were the set-ups of the simulation experiments:

*Occupation Density:* We define the occupation density as the proportion of the scene volume occupied by the subscription and update regions. Greater occupation density results greater probability of matched interest (for the number of objects is constant). In our experiments, the occupation density is set to 1%.

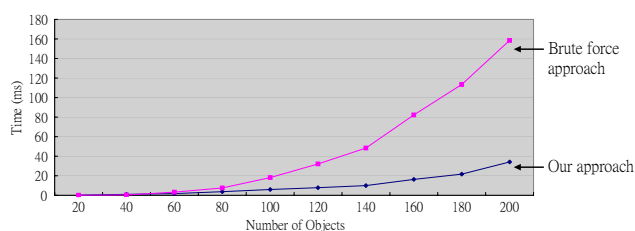
*Object Movements:* Average speed of a moving object equals to 50% of its radius per time step.

*Object Distribution:* The objects are distributed randomly to the scene.

*Number of Objects:* The number of objects varied from 20 to 200 in order to analyze the efficiency for each algorithm.

*Number of Dimensions:* We tested the algorithms in 3-dimensional routing space.

*Number of Regions:* An update region and a subscription region are associated with each moving object.



**Figure 2: Performance comparison of brute force approach and our approach**

The above figure compares the runtime efficiency of the brute force approach and our scalable algorithm. It is not difficult to notice that our approach requires less computational effort for interest matching. This is particularly significant when we gradually increase the number of objects. In summary, our approach imposes less computational overheads and scales better than the brute force approach.

## 5. CONCLUSIONS AND FUTURE WORKS

Interest management is a key technique to address the scalability problem of large-scale CVEs. Prior works concentrated on precise filtering mechanisms, however, that is not enough for real time applications for which runtime performance is an important issue. In this paper we have presented an efficient algorithm for doing interest management in multidimensional routing space. We make use of the existing collision detection algorithm to reduce the workload of interest matching process. Our approach is suitable for applying to HLA DDM service and supports dynamic multicast grouping. Moreover, we have successfully implemented the algorithm in our MOG middleware Lucid Platform 1.0. Experimental results show that our approach scales much better than the brute force matching approach.

Since Lucid Platform 1.0 makes use of single server communication model, the sorting process of interest management we employed is said to be *centralized*. In Lucid Platform 2.0, we will enhance our algorithm to allow distributed processing. Thus, the workload can be shared among multiple servers and further enhance runtime performance.

## 6. REFERENCES

- [1] Institute for Simulation and Training, IST-TR-93-10, Distributed Interactive Simulation Operational Concept [Draft2.2], University of Central Florida, Orlando, Florida, March 1993.
- [2] S. Rak and D. Van Hook, Evaluation of Grid-based Relevance Filtering for Multicast Group Assignment. *In 14th Workshop on Standards for the Interoperability of Distributed Simulations*, pages 739–747, 1996.
- [3] DMSO, Department of Defense, High Level Architecture Interface Specification Version 1.3. 1998.
- [4] MIC, SD, Hong Kong Polytechnic University, Lucid Platform 1.0, <http://www.lucidplatform.com>, 2005.
- [5] J. Calvin, A. Dickens, R. Gaines, P. Metzger, D. Miller, and D. Owen. The SIMNET Virtual World Architecture. *In IEEE Virtual Reality Annual International Symposium*, pages 450–455, 1993.
- [6] J. D. Cohen, M. C. Lin, D. Manocha, and M. K. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. *In Symposium on Interactive 3D Graphics*, pages 189–196, 218, 1995.
- [7] R. M. Fujimoto. *Parallel and Distributed Simulation Systems*. John Wiley and Sons, Inc., 2000.
- [8] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. *SIGGRAPH 1996 Proc.*, 1996.
- [9] C. Greenhalgh and S. Benford. Massive: a collaborative virtual environment for teleconferencing. *ACM transactions on Computer Human Interactions*, 2(3):239–261, September 1995.
- [10] J. Klosowski, M. Held, J. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Trans. on Visualization and Computer Graphics*, 4(1):21–36, 1998.
- [11] M. R. Macedonia, M. J. Zyda, D. R. Pratt, D. P. Brutzman, and P. T. Barham. Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments. *In Virtual Reality Annual International Symposium*, pages 2–10, 1995.
- [12] K. Morse, L. Bic, M. Dillencourt, and K. Tsai. Multicast grouping for dynamic data distribution management. *In 31st Society for Computer Simulation Conference (SCSC '99)*, 1999.
- [13] K. Morse and M. Petty. High Level Architecture Data Distribution Management migration from DoD 1.3 to IEEE 1516. *Concurrency and Computation: Practice and Experience*, 16(15):1–17, 2004.
- [14] K. Morse and J. S. Steinman. Data distribution management in the HLA, multidimensional regions and physically correct filtering. *In 1997 Spring Simulation Interoperability Workshop*, March 1997.
- [15] B. Naylor, J. Amanatides, and W. Thibault. Merging BSP trees yields polyhedral set operations. *Computr Graphics (SIGGRAPH '90 Proc.)*, 24:115–124, 1992.
- [16] M. H. Overmars. Point location in fat subdivisions. *Information Proc. Letters*, 44(5):261–265, 1992.
- [17] H. Samet. *Spatial Data Structures: Quadtree, Octrees and Other Hierarchical Methods*. Addison-Wesley Publishing Company, 1989.