An Experimental Platform for Data Management in Distributed Simulation

Gary Tan, Rassul Ayani, YuSong Zhang and Farshad Moradi
School of Computing
National University of Singapore
Singapore 119260
Corresponding Email: gtan@comp.nus.edu.sg
Fax: (+65) 779 4580

ABSTRACT: In the High Level Architecture (HLA) paradigm, the Runtime Infrastructure (RTI) is similar to a distributed operating system for the simulation federates. The RTI provides a set of services, such as data distribution and management (DDM) among federates. Each federate may inform the RTI about its intention to publish some data, or it may subscribe to receive a subset of the published data. For example, in war gaming a tank may regularly publish its position and another object, e.g., an aircraft, may be interested to detect the position of the tank by subscribing to receive the position update data published by the tank.

The goal of the DDM module in RTI is to make the data communication more efficient by sending the data only to those federates who need the data, as opposed to the broadcasting mechanism employed by Distributed Interactive Simulation (DIS). The approaches used by DDM are aimed at reducing the communication over the network, and the data set required to be processed by the receiving federates.

Efficient data distribution is an important issue in large scale distributed simulations with several thousands of entities. The broadcasting mechanism employed in Distributed Interactive Simulation (DIS) standards generates unnecessary network traffic and is not suitable for large scale and dynamic simulations. An efficient data distribution mechanism should filter the data and forward only those data to federates who need the data.

Several filtering mechanisms have appeared in the literature and some of them have been implemented in RTI. In this paper, we will briefly describe these mechanisms. We describe the design and implementation of a simulation platform used to investigate one of these mechanisms, the grid-based filtering mechanism, and report on the initial experimental results. We show that the size of the grid cells has a substantial impact on the performance of distributed simulations. This platform will serve as a tool for further experimentation on grid-based and other filtering mechanisms.

Keywords: distributed simulation, HLA, RTI, DDM, grid-based filtering, simulation platform

1. Introduction

In the High Level Architecture (HLA) paradigm, the Runtime Infrastructure (RTI) is similar to a distributed operating system for the simulation federates. The RTI provides a set of services, such as data distribution and management (DDM) among federates [1]. Each federate may inform the RTI about its intention to publish some data, or it may subscribe to receive a subset of the published data. For instance, in war gaming a tank may regularly publish its position and another object, e.g., an aircraft, may be interested to detect the position of the tank by subscribing to receive the position update data published by the tank.

The goal of the DDM module in RTI is to make the data communication more efficient by sending the data only to those federates who need the data, as opposed to the broadcasting mechanism employed by Distributed Interactive Simulation (DIS). The approaches used by DDM are aimed at reducing the message traffic over the network, and also reducing the data set required to be processed by the receiving federates. Several filtering mechanisms have appeared in the literature and some of them have been implemented in RTI [2, 3, 4].

In this paper, we will briefly describe these mechanisms. The main aim of this paper is to describe the design and implementation of a simulation platform used to investigate one of these mechanisms, the grid-based filtering mechanism. Preliminary simulation results show that the size of the grid cells has a substantial impact on the performance of distributed simulations.

The rest of the paper is organized as follows. In section 2, we review some of the data filtering mechanisms used in HLA. In section 3 we describe a simulation platform developed with an underlying grid-based filtering mechanism. Section 4 describes the experiments used to investigate the impact of grid-cell size on performance, and discusses the results. The conclusion is presented in section 5.

2. Filtering Mechanisms

A federate that subscribes to an object class’ attribute values (or to an interaction class) will receive all updates of the specified attribute values for all objects of that class. This type of filtering is called class-based filtering and has the benefit of eliminating delivery of attributes of the class (e.g., position of tanks) to those who have not subscribed for it. However, in a class-based filtering a federate must subscribe to the whole class (e.g., all tanks) and not to a subset of it, e.g., all red tanks. The class-based filtering approach may be acceptable for a small federation with few objects in each class. However, for many simulations with large number of objects in each class a better filtering mechanism is needed. The DDM services also provide value-based (or content-based) filtering where a
federate may subscribe to a specific value of the attributes. For instance, using value-based filtering, an aircraft may subscribe only to those tanks that are within the range of its sensors (e.g., those that are located less than 10km from it).

The *region-based* filtering method uses a fundamental construct used by DDM called the routing space (RS). A routing space is a multi-dimensional coordinate system through which a federate may express its interest in receiving data (subscription region) or declares its intention to send data (update region). When an update region and subscription region of different federates overlap, the RTI establishes communications connectivity between the publishing and subscribing federates. Figure 1 shows one subscription region (S1) and two updates region U1 and U2. In this example S1 and U1 overlap and the attribute updates from the object associated with U1 will be routed to the federate that created S1. However, S1 and U2 do not overlap and thus U2’s attributes will not be routed to S1.

![Figure 1: region-based filtering](image)

In figure 1, the federate associated with S1 should receive updates of the object associated with U1 as long it remains in the shaded area. However, in region-based filtering S1 receives all updates of federates within U1. Thus, S1 receives some irrelevant data (those within U1, but outside of the shaded area) as well. To reduce the amount of irrelevant data, the *grid-based* filtering has been proposed.

In grid-based filtering, the area where the data are published, i.e. the routing space, is partitioned into a grid of cells. A grid is used to efficiently determine which part of the subscription region overlaps with which part of the update region. For instance, in figure 2, U1 and S1 overlap in one cell only (cell C22) meaning U1 will publish in this cell and S1 is interested to receive the published data. As opposed to the region-based filtering the data published in cell C21 by U1 will not be sent to S1. Hence, grid-based filtering reduces the amount of unnecessary communications. However, the data published by U1 in C22, but outside the shaded area will also be sent to S1, though S1 does not need it. Nevertheless, the amount of irrelevant data communicated in the grid-based filtering is generally less than the irrelevant data communicated in region-based filtering, but it depends on the grid cell size, as will be discussed later.

![Figure 2: grid-based filtering](image)

Grid-based filtering is achieved by associating two lists with each grid-cell: (i) a list of those objects that fall within the cell at a certain point in time (list of publishers) and, (ii) a list of objects that are interested to receive data about objects of within the cell (list of subscribers).

Obviously, the lists associated with each cell are very dynamic, since the objects are moving and their positions and subscription areas will be changing. Thus, our main concern is *how to communicate efficiently among these very dynamic groups*. An important issue is the size of the grid cells. Larger cell size will produce larger multicast groups associated with each cell, while smaller cell size produces smaller list but it requires more frequent updating of the group lists. Hence, there should be an optimal size for the grid cells. The next section describes an experimental simulation platform used to study the effect of varying the grid-cell size on the performance costs and also to determine the optimal cell size for grid-based filtering schemes.

Several researchers have worked on grid-based filtering, including Cohen and Kemkes [5, 6], Hook and his colleagues [7] and Rizik et al [8]. Cohen and Kemkes [5, 6] discuss the impact of the update/subscription rate on performance of DDM. Rak and Hook [7] study the performance of grid-based filtering algorithms and show the impact of grid cell size on communication costs. Rizik et al [8] use a predator-prey model to identify the impact of cell size on performance of DDM.

### 3. The Simulation Platform

To investigate grid-based filtering, a simulation platform was designed and implemented. The platform will enable us to simulate various synthetic scenarios while changing several parameters. In particular, we want to investigate the effects of varying the grid-cell size in grid-based filtering on the performance of the simulation. The platform will simulate a synthetic scenario of a 2-dimensional routing space containing a number of federates, with each federate having a number of dynamic objects, e.g. tanks in a battlefield, travelling at constant speeds in various directions. Each object has its own update region and subscription region. The battlefield is subdivided into grid cells, and the objects are initially placed at random in the battlefield. The next sub-sections will describe the
architecture, model and algorithm employed in the platform.

3.1 Architecture

The simulation platform is written in the object-oriented programming language C++ and runs on a Fujitsu AP3000 distributed-memory 32-node system. The platform comprises three sub-models:

- SimObj
- DDM manager (Local DDM Manager and DDM co-ordinator)
- Communication Layer

The structure of these three sub-models is shown in figure 3. In this platform, the federate is interpreted as a logical grouping of objects, and is thus not shown in the figure.

SimObj

In this platform, the objects are simulated using the SimObj sub-model. This sub-model is responsible for the movement of the simulation objects within the routing space.

DDM Manager

The DDM manager is in charge of the data filtering strategy in this system and is made up of the co-ordinator and the local DDM managers. The local DDM manager exists in every federate model. The multicast group is decided by the DDM co-ordinator and is sent back to the publisher’s local DDM manager, which will then connect with the subscriber’s local DDM manager and send the data to the subscribers directly. The DDM co-ordinator divides the whole routing space into many cells, and each cell will keep two lists, one is the publishers’ list (Plst), the other is the subscribers’ list (Slst). There is no direct matching between the publishing regions and the subscription regions; both the publisher and the subscriber only register their interests in the related cells’ Plst or Slst by comparing their areas with the cells in the routing space separately, which is called “mapping”.

DDM Coordinator

Figure 3: The DDM Architecture

Figure 4: Grid-based DDM

For example, in a time-stepped simulation and referring to figure 4 above:

At time $t_0$, say, the subscriber S registers its interest into $\{C_{13}, C_{14}, C_{23}, C_{24}, C_{33}, C_{34}\}$

At time $t_0 + dt$ (a time-step later), the publisher U registers its data into $\{C_{32}, C_{33}, C_{42}, C_{43}\}$. So at time $t_0 + dt$, the DDM will maintain a data structure which looks like this:

$\begin{array}{c|c|c|c|c|c}
C_{11} & C_{12} & C_{13} & C_{14} & C_{15} \\
C_{21} & C_{22} & C_{23} & C_{24} & C_{25} \\
C_{31} & C_{32} & S & C_{34} & C_{35} \\
C_{41} & C_{42} & C_{43} & C_{44} & C_{45} \\
\end{array}$

At time $t_0 + dt$, both the publishing region and the subscription region have an overlap in $C_{33}$. In the next time step the DDM manager will notify the publisher (U) to send its updating attributes to the subscriber (S).

In grid-based DDM, a federate may receive a message that it is not interested in. For instance in figure 4, if U and S were detected in $C_{33}$ but did not overlap in cell $C_{33}$, S will still receive the messages published by U in $C_{33}$, though S is not really interested in these messages. Such messages are referred to as “irrelevant messages”. The irrelevant messages are due to the fact that the unit of data sharing is a cell and a federate cannot subscribe to a subset a cell.

In this model, we consider two categories of cost: the cost of updating and management of the publishers’ and subscribers’ lists and the cost of sending irrelevant messages. These costs will depend on the system such as the interconnection network and CPU speed.

Communication Layer

This sub-model is responsible for communication between the federates and the DDM manager. In this layer, the message cost must be considered, which includes the message passing cost to receivers and the irrelevant message passing cost.
3.2 The Algorithm

Figure 5 shows the algorithm incorporated in the platform for performing the time-stepped simulation of the battlefield.

![Algorithm Diagram](image)

**Figure 5: a time-stepped distributed simulation algorithm using grid-based filtering**

4. Scenarios and Experiments

For the purpose of our experiments, we will use two scenarios. The first one will be of a federation of moving tanks within a 40x40km routing space, and the second one will be of an AWACS sensing aircraft flying with several other aircraft.

In scenario 1, two federates, each containing a number of tank objects, are simulated. It is assumed that the tanks move with a constant velocity and constant direction for several time-steps. The tanks are placed at random in the routing space, and their directions are also determined at random (North, South, East or West). For simplicity, we assume that the tanks will turn back upon encountering the battlefield boundary.

This scenario is dynamic in the sense that the objects dynamically modify their update and subscription regions.

In scenario 2, as described in [5], there are several aircraft and an AWACS. The AWACS is circling over the entire area and at each point of time its radar range covers a portion of the routing space (denoted by R1 in Fig 6). The aircraft are circling over the entire area, and are occasionally discovered by the AWACS. In this scenario, the aircraft publish their position (but do not receive any data), while the AWACS only subscribes to the aircraft data (but does not publish anything). This scenario illustrates a dynamic system where the objects are moving very fast and at different speeds.

Simulations were performed using both scenarios. The simulations were run on a Fujitsu AP3000 multiprocessor using the SunOS 5.5.1. Each simulation was carried out for 100 1-minute time-steps.

The first set of experiments involving scenario 1 had two federates, with each federate having twenty tanks in a 40x40km routing space. Each tank has an update region of 1km (i.e. a square of 2kmx2km), and a subscription region of 5km (square of 10kmx10km). The number of grid cells in the routing space was varied and the results are tabulated in table 1.

In the table, the first column (g) refers to the grid cell (square in shape) size in km, and the second column gives the consequent total number of cells in the routing space. The fourth and fifth columns refer respectively to the number of publishing updates and subscription updates per object per time-step of the simulation. Column 6 gives the number of irrelevant messages per object as discussed in section 3.

The total overhead costs for each object per time-step is then given in column 3, and is calculated using the following formula:

\[
\text{Overhead} = c_1 \times (\text{publishing updates} + \text{subscription updates}) + c_2 \times \text{(no. irrelevant messages)}
\]

Where \(c_1\) = cost of a publishing or subscription update, and \(c_2\) = cost of sending an irrelevant message.

The results are also shown as a graph in figure 7, where LUC and IMC stand for List (both publishers’ and subscribers’) Update Costs and Irrelevant Messages Cost respectively.

In the second set of experiments, scenario 2 was used with one AWACS radar and 40 aircraft in a routing space of 600kmx600km square. A publishing distance of 15 km and a subscription distance of 30 km were used. The speed of the aircraft was fixed at 900km/hr and the speed of the AWACS is half of that speed. The results are presented graphically in figure 8.
<table>
<thead>
<tr>
<th>g(km)</th>
<th>No. of cells</th>
<th>SimCost (ms)</th>
<th>Publish Updates</th>
<th>Subscrip Updates</th>
<th>Irrelevant messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.20</td>
<td>(200*200)</td>
<td>473.05</td>
<td>101.10</td>
<td>371.25</td>
<td>0.14</td>
</tr>
<tr>
<td>0.40</td>
<td>(100*100)</td>
<td>135.24</td>
<td>28.03</td>
<td>106.56</td>
<td>0.13</td>
</tr>
<tr>
<td>0.445</td>
<td>(90*90)</td>
<td>111.17</td>
<td>23.86</td>
<td>86.16</td>
<td>0.23</td>
</tr>
<tr>
<td>0.50</td>
<td>(80*80)</td>
<td>80.39</td>
<td>17.64</td>
<td>61.90</td>
<td>0.17</td>
</tr>
<tr>
<td>0.572</td>
<td>(70*70)</td>
<td>67.55</td>
<td>15.17</td>
<td>51.13</td>
<td>0.25</td>
</tr>
<tr>
<td>0.667</td>
<td>(60*60)</td>
<td>52.73</td>
<td>11.46</td>
<td>39.8</td>
<td>0.29</td>
</tr>
<tr>
<td>1.00</td>
<td>(50*50)</td>
<td>37.64</td>
<td>8.44</td>
<td>27.70</td>
<td>0.30</td>
</tr>
<tr>
<td>1.334</td>
<td>(30*30)</td>
<td>25.82</td>
<td>5.68</td>
<td>17.99</td>
<td>0.43</td>
</tr>
<tr>
<td>2.00</td>
<td>(20*20)</td>
<td>11.51</td>
<td>1.90</td>
<td>4.96</td>
<td>0.93</td>
</tr>
<tr>
<td>4.00</td>
<td>(10*10)</td>
<td>13.41</td>
<td>0.72</td>
<td>1.39</td>
<td>2.26</td>
</tr>
<tr>
<td>5.00</td>
<td>(8*8)</td>
<td>12.92</td>
<td>0.45</td>
<td>0.77</td>
<td>2.34</td>
</tr>
<tr>
<td>10.00</td>
<td>(4*4)</td>
<td>24.41</td>
<td>0.15</td>
<td>0.16</td>
<td>4.82</td>
</tr>
<tr>
<td>20.00</td>
<td>(2*2)</td>
<td>71.68</td>
<td>0.04</td>
<td>0.09</td>
<td>14.31</td>
</tr>
<tr>
<td>40.00</td>
<td>(1*1)</td>
<td>183.15</td>
<td>0.00</td>
<td>0.00</td>
<td>36.63</td>
</tr>
</tbody>
</table>

Table 1: Scenario 1 (no. of tanks=40, Update region=1.0km, subscription Region=5.0km, c1=1ms, c2=5ms)

Analysis

From figures 7 and 8, it can be seen that choosing a smaller cell size will increase the number of position and subscription updates, but will decrease the number of irrelevant messages. On the other hand, a larger cell size will decrease the position/subscription update costs but increase the number of irrelevant messages.

It can also be seen that the overhead of DDM is high with small grid cell size. This overhead decreases with increasing grid cell size until an optimal grid cell size is used, thereafter the cost starts to increase again. Hence, we see that the size of the grid cells has a substantial effect on the performance of the simulations, and there should be an optimal cell size for grid-based filtering.

5. Conclusions

Efficient data distribution is an important issue in large scale distributed simulations with several thousands of entities. The broadcasting mechanism employed in Distributed Interactive Simulation (DIS) standards generates unnecessary network traffic and is unsuitable for large scale and dynamic simulations. In this paper we discussed several filtering mechanisms and grid-based filtering in particular.

We introduced a simulation platform for investigating grid-based filtering, and preliminary results from experiments using the simulation platform on two typical scenarios indicate that the grid cell size has a substantial impact on the performance of distributed simulations.

This platform serves as a convenient tool for investigating the effects of grid-based filtering, and further experiments will have to be conducted, such as varying the subscription regions, and varying the cost factors, to see how these affect the choice of the grid cell size. The platform can also be easily modified to support other filtering mechanisms, and the experimental results can then be used to compare the efficiencies of the various filtering mechanisms.

6. References

2. _HLA Data Distribution Management: Design Documents Version 0.7_ (November 12, 1997).
5. Danny Cohen and Andreas Kemkes, _User-Level Measurement of DDM Scenarios_, in Proceedings of


7. **Author Biographies**

**Gary Tan**

Gary TAN obtained his B.Sc (Hons 1) in Physics from the National University of Singapore in 1984. His M.Sc and Ph.D were in Computer Science from the University of Manchester in 1986 and 1992 respectively. He is currently an assistant professor in the School of Computing at the National University of Singapore and a member of the Modelling and Simulation Group in the Department of Computer Science. His current research interests are parallel and distributed systems, load balancing and scheduling, parallel and distributed simulation, and high level architecture. He is currently on the program committee and publicity chair of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2000).

**Rassul Ayani**

Rassul AYANI is a Professor of computer science at the Department of Teleinformatics, Royal Institute of Technology (KTH), Stockholm, Sweden. Currently, he is a visiting Professor at the School of Computing, the National University of Singapore. His current research interests are in Distributed Systems, HLA, DIS and parallel simulation.

Dr Ayani has been working on parallel and distributed simulation for the past ten years and has published over 60 articles in this area. He and his students at KTH have developed a parallel simulation environment that has been used for simulation of computer architectures, mobile communication systems, and ATM networks/switches.

Dr. Ayani is an associate editor of the ACM Transactions on Modeling and Comuter Simulation (TOMACS) has served as program chair and program committee member at numerous international conferences and is currently program co-chair of the Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS) that will be held in San Francisco CA, USA, Aug 29 - Sep 1, 2000 (For more info on MASCOTS’2000 see http://www.ee.uwa.edu.au/~mascots/).

**YuSong Zhang**

YuSong Zhang received his BS degree from Nankai University, China, in 1994. He is currently a Masters student and research scholar in the Modelling and Simulation Group in the department of Computer Science, School of Computing at the National University of Singapore. His research interest includes parallel and distributed simulation. He will be submitting his M.Sc Thesis shortly.

**Farshad Moradi**

Farshad MORADI is a Research Fellow in the School of Computing at the National University of Singapore and a member of the Modelling and Simulation Group in the Department of Computer Science. He is currently on leave from Swedish Defence Research Establishment (FOA), where he works as Research Officer. He holds a Master of Science in Computer Science and Engineering from Chalmers University of Technology, Gothenburg, Sweden. His research interests are in the areas of distributed systems, modeling and simulation and agent technology.