IMPLEMENTING DISTRIBUTED DATA MANAGEMENT SYSTEM IN DYNAMIC OBJECTS BY USING IMPROVED SORT-BASED ALGORITHM

Nwe Nwe Myint Thein, Swe Zin Hlaing
University of Computer Studies, Yangon
nnmyint.thein@gmail.com,swezinhlaingucsy@gmail.com

ABSTRACT

In the High-Level Architecture (HLA) paradigm, the Runtime Infrastructure (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. DDM services are used to reduce the transmission and receiving of irrelevant data and aimed at reducing the communication over the network. These services rely on the computation of the intersection between "update" and "subscription" regions. Currently, there are several main DDM filtering algorithms. Our proposed system describes data management and filtering mechanism on tank simulation in battlefield area. This system intends to detect the movement of the tank objects, search overlap between the tank object and every regional regiment (extent). When overlapping information is getting from one of the simulation object (OverlapDetector), another simulation object (Coordinator) connects the relevant extent that containing the tank object. That extent continued to send the tank information to other regional regiments according to the predefined list. In this paper, we present the design and implementation of a simulation platform used to implement one of the filtering algorithms, the improved sort-based algorithm, and report the overhead of reducing network traffic and ensuring that the forwarding data receive federates only who need the data .

Key words High Level Architecture, Data Distribution Management, Improved Sort-Based matching algorithm

1. INTRODUCTION

There are several groups of services, which are provided by the HLA Runtime Infrastructure (RTI) to coordinate the operations and the exchanges of data between federates (simulations) during a runtime execution. The interaction of object instances across user applications is supported by the function of RTI,which is similar to a distributed operating system.

Data Distribution Management (DDM) mechanisms are necessary to provide efficient scalable support for large scale distributed simulations. Data distribution management (DDM) intends to limit and control the volume of data exchanged during a simulation and reduce the processing requirements of federates. The key to efficient DDM is to limit the messages sending to include only those messages with state updates intended for federates that require them. The High-Level Architecture (HLA) specifications define six services including DDM services supported by the Runtime Infrastructure (RTI). In a distributed simulation environment, every simulated movement or action that takes place on a simulator, which may affect or interest to another participant in the simulation, require a message. Such actions can include, but are not limited to, the movement, creation or destruction of simulated entities. In a large-scale distributed simulation, simulating many entities across many different federates; the entities that are of interest to other entities can result in increased communication across a network. DDM are aimed at reducing the message traffic over the network and the important information can be distributed to the federates only who need the data. The purpose of this paper is to implement the intersection algorithm for the DDM especially for improved sort-based algorithm. In this paper, we implement this algorithm with two dynamic tank objects, incorporating varied numbers of federates in a battlefield area, this paper offers more insight into the effects of intersection between two tank objects and federates on the efficiency of improved sort-based algorithm. This algorithm within a simulation that detects the position of tank and find out the intersection between the tanks objects and every regional regiment (federates) and also firstly detected regional regiment and the predefined list of regional regiments receive the tank information to make the requirement actions in a timely manner.

The remainder of the paper is organized as follows. Section 2 describes HLA issues relevant to data distribution. The Improved Sort-based Algorithms for DDM matching methods is explained in section 3. Section 4 represents the simulated scenario implemented with an improved sort-based matching algorithm on tank objects in the battlefield area. Section 5 presents the performance analysis of the system. Finally, section 6 offers conclusion and further work.

2. OVERVIEW OF DDM IN THE HLA

Within the Department of Defense (DoD), the HLA provides a common architecture for modeling and simulation. HLA's Run Time Infrastructure (RTI) provides commonly required services to simulation systems that service a distributed operating system

providing to applications. HLA services are grouped into six categories:

- Federation Management (FM)
- Declaration Management (DM)
- Object Management (OM)
- Ownership Management (OWM)
- Time Management (TM)
- Data Distribution Management (DDM)

DM and DDM are used to specify which federates should receive messages for each attribute update and interaction. DM services (Publish and Subscribe) to allow a federate to update and receive updates to object attributes based on object class. The RTI uses information provided in publish/subscribe calls to set up filters that direct data among federates that need them. OM (Update Attribute Values) service call notifies the RTI that one or more attributes have been modified. For example, an object might subscribe to the attribute 'location' of all tanks in the battlefield. However, such filtering will only be appropriate for relatively small federations. DDM services provide more powerful data distribution services enabling value-based filtering. For example, a tank might want to receive data from other tanks only if they are in its visible range. RTI does not know the meaning of object attributes it cannot provide range-based filtering in this example. Federates must agree on a filtering strategy to do this.

The fundamental Data Distribution Management constructs a routing space. A routing space is a named sequence of dimensions, which form a multi-dimensional coordinate system. Regions are defined as sets of extents which are rectangles representing the sensor ranges of different units. Federates either express an interest in receiving data (subscribe) or declare their intention to send data (publish). These intentions are expressed through:

Subscription Region: Bounding routing space coordinates which narrow the scope of interest of the subscription federate.

Update Region: Bounding routing space coordinates which are guaranteed to enclose an object's location in the routing space. Both subscription and update regions can change in size and location over time as a federate's interests change or an object's location in the routing space changes.

An object is discovered by a federate when at least one of the object's attributes come into scope for federate, i.e. if an only if: the federate has subscribed to the attribute of the object's update region overlaps the federate's subscription region. Specifying a subscription region, the federate tells the Run Time Infrastructure (RTI) that it is interested in data which fall within the extents of the region specified by that federate. Specifying an update region which is associating with a particular object instance is a contract from the federate to the RTI. At a time step, that federate will ensure the characteristics of the object instance which map to the dimensions of the routing space falling within the extents of the associated region there by the attribute update is being issued. The federate is monitoring these added characteristics, for each of the attributes owned by the federate. As the state of the objects change, the federate may need to either adjust the extents on the associated regions or change the association to another region. Each federate can create multiple update and subscription regions. Update regions are associated with individual objects that have been registered with the RTI. A federate might have a subscription region for each sensor system being simulated.

3. IMPROVED SORT-BASED MATCHING ALGORITHM

Improved sort-based algorithm is interested in the nonoverlap information. This algorithm needs to know which extents do not overlap with each other. Then on the reverse, their overlap information can be easily known. This algorithm is not complicated. If two extents do not overlap with each other, one extent should fall out of the range of the other extent. There are only two types of subscription extents if a subscription extent does not overlap with the update extent U in one-dimensional space in Figure 1.

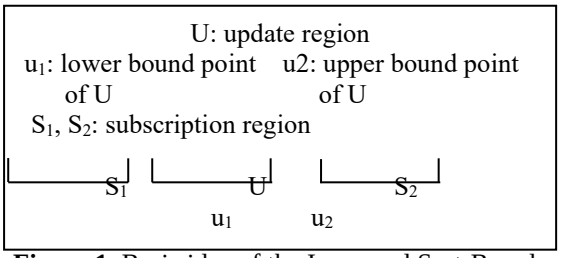

Figure 1. Basic idea of the Improved Sort-Based Algorithm

One type of extent is one whose end points are both located before U, such as S1 (i.e. both points are less than u1). These extents are marked as nonoverlapping with U, when processing u1. The other type of extent is one whose end points are both located after U, such as S2 (i.e. both points are greater than u2). When processing u2, these extents are marked as nonoverlapping with U. All the nonoverlap information of the update extent U can be acquired and the information is stored only at the update extent side. In the improved sort-based algorithm, shown in figure 2, two sets are employed to store the nonoverlap information, The SubscriptionSetBefore set stores the subscription extents located before the current update extent, and the SubscriptionSetAfter set stores those subscription extents located after the current update extent.

In the beginning, it is assumed that all subscription extents are located after the current position since no extent has been processed. As a result, the SubscriptionSetBefore set is nil (line 7), while the SubscriptionSetAfter set is universal set (line 8). When a lower bound point of a subscription extent is processed, the extent is no longer located after the current update extent, and it is removed from the SubscriptionSetAfter set (line 14). When an upper bound point of a

subscription extent is processed, the extent is located before the current update extent. Therefore, it is appended to the SubscriptionSetBefore set (line 16). When a point of an update extent is processed, the nonoverlap information is stored in the SubscriptionSetBefore set, or the SubscriptionSetAfter set accordingly (line 19 and 21).

```
(1)  for each extent Ri in the routing space
(2)  {
(3)  insert lower bound point of Ri into list L
(4)  insert upper bound point of Ri into list L
(5)  }
(6)  sort list L
(7)  SubscriptionSetBefore = ø
(8)  insert all subscription extents into
     SubscriptionSetAfter
(9)  for all point Pi in the sorted list L
(10) {
(11) Ri = Extent Id of Pi
(12) if (Ri is a subscription extent) {
(13) if (Pi is the lower bound point of Ri)
(14) remove Ri from SubscriptionSetAfter
(15) else  // Pi is the upper bound point of
Ri
(16) insert Ri into SubscriptionSetBefore
(17) } else { // Ri is an update extent
(18) if (Pi is the lower bound point of Ri)
(19) all extents in SubscriptionSetBefore
do not overlap with Ri
(20) else
(21) all extents in SubscriptionSetAfter
do not overlap with Ri
(22) }
(23) }
```

Figure 2. Improved Sort-Based Algorithm

The scenario for the improved sort-based algorithm is presented by Figure3.

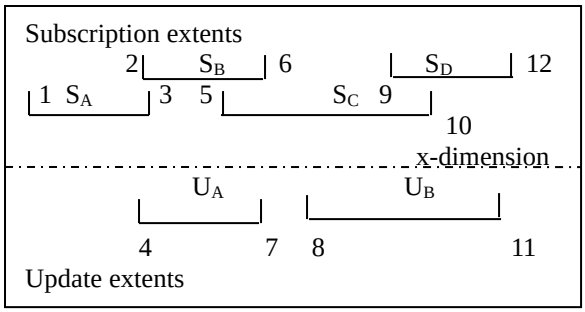

Figure 3. Scenario for the Improved Sort-Based Algorithm

SA1 ----- SA is no longer located after the current or Latter update extents
SA is removed from SubscriptionSetAfter
SB1 -----same as SA1
it is removed from SubscriptionSetAfter
SA2 -----SA is now located before the current or latter update extents
it is inserted into the SubscriptionSetBefore
UA1 -----UA does not overlap with the extents in SubscriptionSetBefore, namely SA
SC1 -----it is removed from SubscriptionSetAfter
SB2 -----append it to the SubscriptionSetBefore set
UA2 -----UA does not overlap with the extents in SubscriptionSetAfter, namely SD

In step 8, UB does not overlap with the extents in SubscriptionSetBefore $\{S_A,S_B\}$.
In steps 9 and 10, at first S_D is removed from SubscriptionSetAfter, and then S_C is appended to SubscriptionSetBefore.
In the last step, U_B should not overlap with extents in SubscriptionSetAfter.
However, since SubscriptionSetAfter is null, no more Nonoverlap information is acquired during this step.
Lastly, U_A does not overlap with S_A or S_D, U_B does not overlap with S_A or S_B. The final overlap information is that U_A overlaps with S_B and S_C, U_B overlaps with S_C and S_D.

4. THE SIMULATION PLATFORM

To implement improved sort-based algorithm, a simulation platform was designed. In particular, the simulation intends to investigate the effects of improved sort-based algorithm on the performance of the simulation. The platform simulates a 2-dimensional routing space containing dynamic objects. E.g. moving tanks in a battlefield, traveling at constant speeds in various directions. The battlefield contains twelve extents within four regions and the two tanks. These tanks are initially placed at user's input coordinates in the battlefield. The next sub-sections describe the architecture and algorithm employed in the platform.

4.1. The Proposed Architecture

The simulation platform is written in the Java programming language and runs on an interconnected network using 5 computers. The platform comprises three sub-models:

- Tank Controller
- Overlap Detector
- Coordinator

The structure of these three sub-models is shown in figure 4.In this platform, the federate is interpreted as a logical grouping of objects, and is shown in the Figure 6.

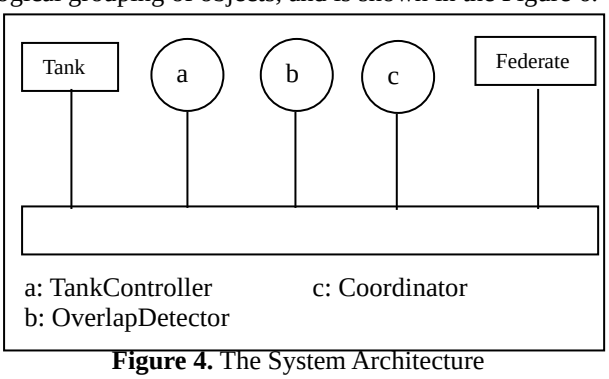

Figure 4. The System Architecture

Tank Controller In this platform, the tank objects are simulated using the Tank Controller sub-model. This sub-model is responsible for the movement of the simulation objects within the routing space and it calculate the congestion point for two objects and control the tank object's movement. When the tank objects see the boundary of the battlefield area or they reach the congestion point, these tanks turn the new direction. The possible new directions are represented in table according to start direction and Figure 5 illustrates the trend for tank object's movement. The tank objects move according to user's assigned direction and at each time step it checks for meeting the boundary of battlefield area or congestion point. When the tanks meet the battlefield's boundary or congestion point, Tank Controller send new direction. So the tanks can turn their appropriate new direction.

Table1. Possible New Directions From Start Direction

Start direction	Possible direction from start position	New direction to turn tank object
East	West	Southwest to Northeast Northwest to Southeast
East	Southwest to Northeast	South
East	Northwest to Southeast	North
West	East	Northeast to Southwest Southeast to Northwest
West	Southeast to Northwest	South
West	Northeast to Southwest	North
South	North	Northwest to Southeast Northwest to Southeast
South	Northeast to Southwest	East
South	Northwest to Southeast	West
North	South	Southwest to Northeast
North	Southwest to Northeast	West
North	Southeast to Northwest	East

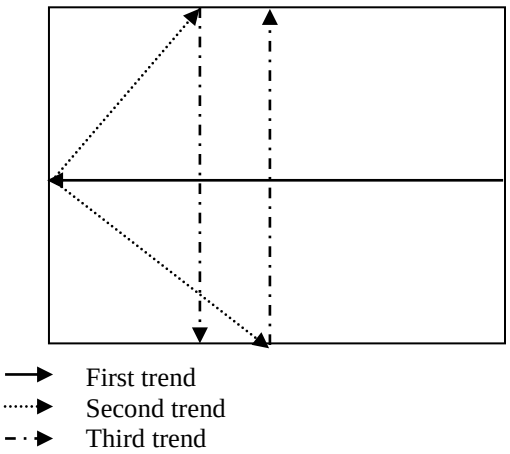

Figure 5.. Tank Object's Movement from East Side of Battlefield Area

Overlap Detector The Overlap Detector is in charge of the data filtering strategy in this system. It calculates the overlap between the tank objects and all the federates within the battlefield area.OverlapDetector store the lower bound extents and upper bound extents of each regional regiment and tank objects, SubscriptionSetBefore and SubscriptionSetAfter. All regional regiments are initially inserted in the SubscriptionSetAfter. Null value is initialized into the SubscriptionSetBefore. If OverlapDetector detects the lower bound point of regional regiment, removes it from SubscriptionSetAfter. If it detects the upper bound point of regional regiment, insert into SubscriptionSetBefore. If OverlapDetector detects the lower bound point of tank objects, it decides all extents in SubscriptionSetBefore do not overlap with tank object, it decides all extents in SubscriptionSetAfter do not overlap with tank object. If it knows the nonoverlap information, it can know the overlap information and retrieve the name of overlapped regiment. When it knows the overlapping information, it connects regional regiment (federate) and other regional regiments which need to know that information. Then it sends the command directly to overlapping federate and the tank information to other regional regiments in the predefined list. In this sub-model, OverlapDetector require to communicate with Coordinator.

Coordinator This sub-model is responsible for communication between the federates and the DDM manager. In this sub-model, the message cost must be considered, which includes the message passing cost to receivers and the replying message cost to the sender.

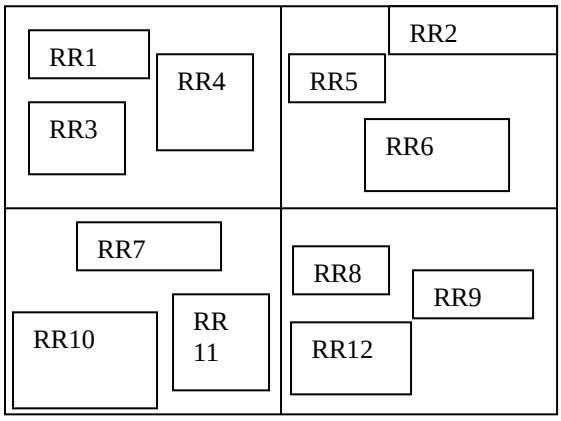

Figure 6. The Battlefield Area

4.2. The Proposed Algorithm

Figure7. shows the algorithm incorporated in the platform for performing the time-stepped simulation of the battlefield.

```
Start simulation
Initialization
Initialize the location of the tank objects(x,y coordinate), choose start direction and type possible direction
End-initialization
For t=t0, t0+dt, t0+2dt, ….. , do
        begin
        -TankController captures the movement of tanks and sends it to RTI
        -OverlapDetector calculate the overlapping between tank objects and regional regiments
        -RTI DDM determines subscription region and inform the tank information
        -Coordinator connect between the actual overlapping subscriber and other extents (regional regiments) according to the predefined list
        -each subscriber received data and makes require action
        end
endfor
```

Figure 7. A tTime-Stepped Distributed Simulation Algorithm Using Improved Sort-Based Matching Algorithm

5. PERFORMANCE ANALYSIS OF THE PROPOSED SYSTEM

5.1. Theoretical Performance Analysis

In the improved sort-based algorithm, it is assumed that, there are N subscription extents and N update extents in a routing space. According to the algorithm, there are four main steps.

Sorting the list according to the coordinates. There are 2*(N+N) coordinates for each dimension, initializing and constructing the list for each dimension is O(n). The complexity of sorting the list is O(n log n) using heap sort. Initializing and constructing two bit vectors is O(n).
Porcessing the points of subscription extents. For each dimension, there are N iterations to insert the extents in the SubscriptionSetBefore set and another N iterations to remove the extents from the SubscriptionSetAfter set. The data structure of bit vector is employed, inserting or removing an element only requires constant time. The complexity of these operation is O(n).

Processing the points of update extents. For each dimension, there are, in total, 2*N iterations to transfer the information from two sets (SubscriptionSetBefore and SubscriptionSetAfter) to the array of bit vectors used to store the overlap information. There are N elements in each bit vector, the complexity of this operation is O(n2).
Combining the result of each dimension. Finally, bitwise 'AND' operations are used to get the overall overlapping information of all dimensions. The complexity of this process is O(n2).

The overall complexity of the improved sort-based algorithm is quadratic. In this algorithm, many bit operations have been utilized. The performance of the DDM services is related to several parameters including (1) the processing cost of the matching algorithm, (2) the transmission cost of the network, (3) the total filtering cost of the irrelevant messages.

5.2. Factors Affecting Performance

There are three factors affecting the performance. These are:
Total number of extents: If there are more extents, it is to cost more time to get the matching result. In the experiment, the total number of extents (update extents plus subscription extents) is varied from 12 to 60, and the numbers of subscribers are six times of the number of updaters. All of the extents are uniformly distributed in the routing space.

Number of extents in each region: If all the extents belong to the same region, much less information needs to be returned than if each extents was in a different region. In the experiments, each region has the same number of extents and it may have 3, 6, or15 extents.
Overlapping degree: The overlapping degree is also an important factor. In improved sort-based approach needs to check the coordinates of two dimensions. It is interested in the nonoverlap information. This algorithm needs to know which extents do not overlap with each other. Then on the reverse, their overlap information can be easily known.

In this paper, number of extents in each region is not important factor, because the system intends to detect the overlap information between the total number of subscription extents and the number of update extents (tank objects). In this system, the processing cost of the matching algorithm and the transmission cost of the network are included. But the total filtering cost of the

irrelevant messages are not included because the actual overlapping federate are produced by improved sort-based algorithm and the other federates which need to know the tank information are predefined. These federates (Regional Regiments) are informed of the tank information. The network connection cost depends on the total number of subscription extents to send the tank information between the subscription extents.

Figure 4 shows the overhead cost of the various numbers of the subscription extents and update extents (update regions) including network connection cost.

overhead = (c * subscription updates) + n

where c = cost of a subscription update, and

n = network connection cost

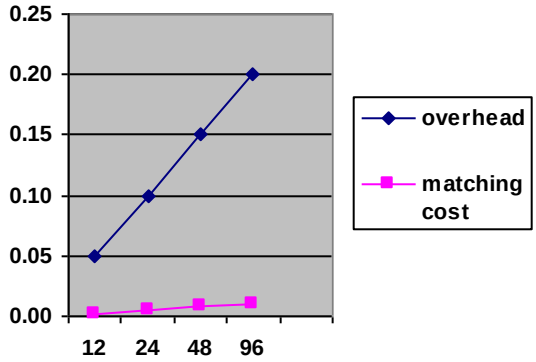

Figure 8. Performance of the Improved Sort-Based Algorithm Based on Total Number of Extents

6. CONCLUSION

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 are interested in improved sort-based matching algorithm in particular. A simulation platform implement data distributed management in dynamic tank objects by using improved sort-based matching algorithm. This platform serves as a convenient tool for implementing the improved sort-based algorithm in data distribution management. Because this system does not need the overlap relationship is symmetrical and the algorithm to store overlap information both at the subscription extent side, and at the update extent side. Improved sort-based algorithm records the intersection information only at one type of extent side. The system also does not include the filtering cost of the irrelevant messages. The platform can also be easily modified to support other matching algorithm and the experimental results can then be used to compare the efficiencies of the various matching mechanisms.

REFERENCES

[1] A. Boukerche, C. Dzermajko. “Dynamic Grid-Based vs. Region-based Data Distribution Management Strategies in Multi-Resolution Large-Scale Distributed System” , PDP’03: *Proceedings of the 17th International Symposium on Parallel and Distributed Processing, Nice , France*(2003 April)

[2] C. Raczy, G. Tan, J. Yu. “A Sort-Based DDM Matching Algorithm for HLA” , *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, Volume 15 Issue 1,2005, January

[3] F. Moradi, G. Tan, R. Ayani, Y. S. Zhang. “*An Experimental Platform for Data Management in Distributed Simulation*”, http://www.siaa.asn.au/get/2395365387.pdf

[4] I. Tacic, R. T. Fujimoto, “Synchronized Data Distribution Management in Distributed Simulations”, *Proceedings Twelfth Workshop on Parallel and Distributed Simulation*, 1998