SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

September 2012 Vol. 55 No. 9: 2090–2101
doi: 10.1007/s11432-011-4427-0

Displacement residual based DDM matching algorithm

ZHANG Lin[1,2], ZHOU Zhong[1,2]*, LIU Lin[1,2] & WU Wei[1,2]

[1]*State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing* 100191*, China;*
[2]*School of Computer Science and Engineering, Beihang University, Beijing* 100191*, China*

Received December 7, 2010; accepted May 30, 2011; published online April 12, 2012

Abstract High level architecture (HLA) is the prevailing standard for modeling and simulation. The data distribution management (DDM) service of HLA is defined for reducing the delivery of irrelevant data. The key in DDM implementation is the region overlap computation, i.e. the matching between update and subscription regions. Existing algorithms usually make a compromise between region fidelity and network payload. This paper takes both the matching algorithm efficiency and bandwidth cost into account. The main contributions are: 1) illustrating the relationship between region changes and overlap changes, as helps reduce the number of region matching and then improves the total matching efficiency; 2) classifying region updates into two types of data expression, snapshot and residual. The network traffic will be reduced by transmitting only residual data instead of full region representations occasionally. Consequently, a region matching algorithm called displacement residual-based DDM matching (DRBM) is proposed in the paper. Theoretical analysis, algorithm implementation and experiment evaluation are presented. Experiment results show that DRBM provides better matching performance and significant network payload reductions especially when there is a large number of changing regions.

Keywords distributed simulation, high level architecture, data distribution management, region snapshot, region residual

Citation Zhang L, Zhou Z, Liu L, et al. Displacement residual based DDM matching algorithm. Sci China Inf Sci, 2012, 55: 2090–2101, doi: 10.1007/s11432-011-4427-0

1 Introduction

In a large-scale distributed simulation, thousands of objects keep moving and interacting in a virtual environment. Because each object is producing messages nearly all the time, a data explosion results that cripples the simulation performance and restricts its scalability when real-time interaction is required. Interest management technologies are used to reduce message traffic over the network, mostly according to the spatial relationship of sender/receiver objects. Researchers have been studying interest management technologies since the 1990s. High level architecture (HLA) is the prevailing standard for modeling and simulation. It regulates a region-based interest filtering service group called data distribution management (DDM). Three main concepts, routing space, update region, and subscription region, are defined in DDM services in the HLA 1.3 standard [1–4]. DDM services allow producers of data to assert properties of

*Corresponding author (email: zz@vrlab.buaa.edu.cn)

their data, and consumers of data to specify their data requirements in terms of user-defined regions. The run-time infrastructure (RTI) then distributes data from producers to consumers based on matches between these update and subscription regions. Hence, the region-matching algorithm becomes critical in judging whether regions overlap. Indeed, DDM services constitutes a problem in regard to algorithm efficiency optimization and in delivering region information. Both algorithm efficiency and region delivery are very important in any practical implementation of DDM. There exist four main DDM matching algorithms classified by approach: region-based [5], grid-based [6–8], hybrid [9–14], and sort-based [15–17]. Each has its respective advantages and disadvantages, but none take network delivery into account. This paper investigates the bound limit of sort-based matching, determins the relationship between region-changes and overlap-changes, and then proposes the displacement residual-based DDM matching (DRBM) algorithm, which further reduces the matching scope and improves efficiency. In regard to the region exchange problem, the region update is attached to a type of snapshot or residual. Similar to the idea of inter-frame image compression, an estimation region update can be deduced from a residual after referring to the latest snapshot region update. Region delivery performed in this manner can be established at a relatively lower bandwidth cost.

2 Overview

In this section, the four main DDM matching algorithms are described highlighting their respective advantages and disadvantages.

The region-based approach is a brute-force approach that checks every pair of subscription region and update region to obtain exact overlapping information [8]. The computational complexity of region-based approach is $O(N^2)$.

To reduce this complexity, a grid-based approach was proposed that divides the routing space into a grid of cells. Each region is then mapped to grid cells. An update region and a subscription region are assumed to overlap with each other if and only if these regions share at least one common grid cell. However, the grid-based approach cannot derive accurate overlapping information. Hence irrelevant data may be received by each receiving federate [18,19].

Hybrid methods have been proposed because grid-based and region-based approaches have completely contradicting features related to filtering efficiency [20,21]. They use the grid-based approach to reduce the number of regions needing to be matched and the region-based approach to obtain further filtering results [13]. In this way, the matching computational complexity is reduced compared with the pure region-based approach and the derived overlapping information is exact. The major problem is that its performance depends on the chosen size of grid cells.

The sort-based approach, proposed in [17], first inserts the bounds from all regions into a list L and sorts L, next statistically processes all the regions in one round, and then derives all the overlapping information. The performance of sort-based approaches is good due using bit operations, but major drawbacks remain. In large scale environments it is not practical because of its quadratic storage increase. Furthermore, if a certain region changes, the algorithm must process all regions once again at great computational cost [16].

Pan proposed an improved sort-based approach in [16]. A search is restricted to a certain sub range based on a given necessary and sufficient condition for region overlapping. In this approach, distinct from the original sort-based approach, the upper and lower bounds of all subscription regions are separately sorted per dimension, as well as update regions. Then for a specific region, to derive its overlapping information, it separately scans in a sub range on the upper-bound list and lower-bound list per dimension. Matching performance is good, especially in large spatial environments, but shortcomings are obvious as well. Because, if there are large number of entities with relatively small interest regions and a small number of entities with relatively big interest regions in the spatial environment scanning within the maximum possible extent for each range usually consumes computing resources.

The DRBM algorithm proposed in this paper has some distinctive differences from existing sort-based algorithms. First, region scanning is further shortened and proofs of the related theorems are given. The

algorithm efficiency is decided mainly on the matching efficiency, so scanning of a smaller region will help. Second, network delivery is taken into account whereas none of the previous work has done so.

3 Bounds in region matching

To understand the characteristics of region overlap, a simple scenario is examined. An example is shown in Figure 1 to illustrate region overlapping among three regions in a two-dimensional coordinate space. Region B overlaps both region A and C, while region A has no intersection with region C.

If regions are projected onto one common dimension, as depicted in Figure 1, all overlapping information in that dimension can be computed. Two regions overlap if and only if they have common dimensions and their ranges overlap on each common dimension. Hence, matching detection in a multi-dimensional routing space can be carried out dimension-by-dimension [17]. Each routing space may have one or more dimension, each representing a specific characteristic of the routing space. There is no restriction on the number of dimensions in both HLA 1516 and earlier HLA 1.3 standards.

Thus, it is necessary to analyze overlapping cases among ranges of regions in one common dimension. As shown in Figure 2, there are four possible overlaps between ranges $u_i[u_{i,l}, u_{i,u})$ and $s_j[s_{j,l}, s_{j,u})$ in a common dimension. Thus, the necessary and sufficient condition for two ranges to overlap is the satisfaction of both inequalities 1 and 2 [16].

$$u_{i,u} > s_{j,l}, \tag{1}$$

$$s_{j,u} > u_{i,l}. \tag{2}$$

Therefore, to derive exact overlapping information for a specific update range u_i, all subscription ranges which satisfy both 1 and 2 should be checked. Computational cost could be high if all subscription ranges in this dimension are checked each time, especially in large routing spaces. In this paper, we first investigate how to shorten the matching process.

For ease of explanation, overlap characteristics between an update range u_i and all subscription ranges in a common dimension are given next. In regard to the subscription range, all characteristics are the same.

We first formalize the basic matching problem.

We denote the set of all subscription regions by Φ and the set of all update regions by Ψ. In dimension d, let ϕ^d and ψ^d be the respective sets of all subscription and update ranges. Given an update region U_i, we let ϕ_i denote the set of all subscription regions overlap U_i. Similarly, let ψ_i denote the set of all update regions overlap the subscription region S_i.

Let ϕ_i^d be the set of subscription ranges overlap the update range u_i in dimension d, then $\phi_i^d = \{s_j | s_j \text{ overlaps } u_i, s_j \in \phi^d\}$. Likewise, let ψ_i^d be the set of update ranges overlap the subscription range s_i in dimension d, then $\psi_i^d = \{u_j | u_j \text{ overlaps } s_i, u_j \in \psi^d\}$.

Definition 1 (maxSRS, maximum subscription range size)**.** For a given dimension d, maxSRS refers to one simplified form of the maximum range size in ϕ^d. maxSRS = max {sizes of all the subscription ranges in ϕ^d}.

Definition 2 (maxURS, maximum update range size)**.** For a given dimension d, maxURS refers to one simplified form of the maximum range size in ψ^d. maxURS = max {sizes of all the update ranges in ψ^d}.

Theorem 1. $\forall s_j \in \phi_i^d$, we have $u_{i,l} - \text{maxSRS} < s_{j,l} < u_{i,u}$ and $u_{i,l} < s_{j,u} < u_{i,u} + \text{maxSRS}$.

Proof. From formula 1, we have: $s_{j,l} < u_{i,u}$. From formula 2, we have: $s_{j,u} > u_{i,l}$, then $s_{j,u} - s_{j,l} > u_{i,l} - s_{j,l}$, $s_{j,l} > u_{i,l} - (s_{j,u} - s_{j,l})$. With Defination 1, we thus have: $s_{j,l} > u_{i,l} - \text{maxSRS}$. We thus have: $u_{i,l} - \text{maxSRS} < s_{j,l} < u_{i,u}$. Similarly, we can prove: $u_{i,l} < s_{i,u} < u_{i,u} + \text{maxSRS}$.

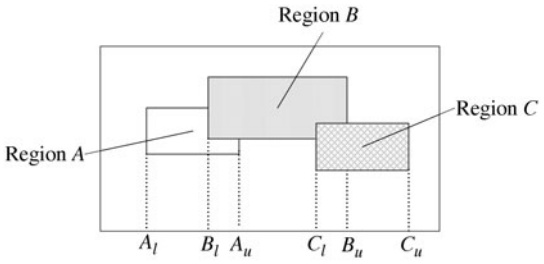

Figure 1 Example of region overlapping.

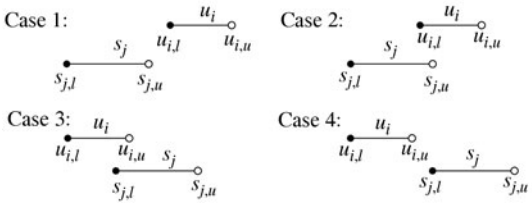

Figure 2 Overlapping cases for two ranges.

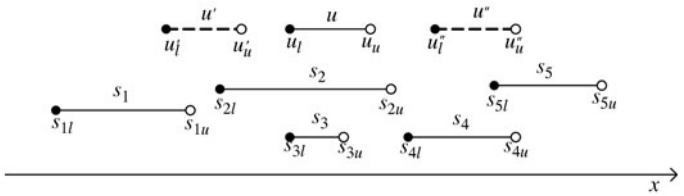

Figure 3 Changes in overlap caused by modification of a range.

Theorem 2. $\forall u_j \in \psi_i^d$, we have $s_{i,l} - \text{maxURS} < u_{j,l} < s_{i,u}$ and $s_{i,l} < u_{j,u} < s_{i,u} + \text{maxURS}$.

The proof of Theorem 2 is similar to that of Theorem 1.

As mentioned before, frequent movements of objects bring about frequent modifications of their regions. To investigate changes in overlap caused by these modifications, a simple scenario is examined. Figure 3 shows one update range u and five subscription ranges along dimension X. The u' and u'' are two examples of modifications of u.

A modification of a range can be viewed as a result of the movement of its two bounds. As shown in Figure 3, when the lower bound of u (u_l) moves across the upper bound of s_1 ($s_{1,u}$) to u'_l, an overlap between u and s_1 emerges. Similarly, when u_u moves across $s_{3,l}$ to u'_u then, overlap between u and s_3 disappears. When u_l moves across $s_{2,u}$ and $s_{3,u}$ to u''_l, u no longer overlaps s_2 and s_3. When u_u moves across $s_{4,l}$ and $s_{5,l}$ to u''_u then, u overlaps s_4 and s_5.

From the above, we get four important corollaries. Given an update range u_i and a subscription range s_j, we have:

Corollary 1. If $s_{j,l}$ moves across $u_{i,u}$ in negative direction with respect to the dimension, an overlap between s_j and u_i emerges.

Corollary 2. If $s_{j,l}$ moves across $u_{i,u}$ in positive direction with respect to the dimension, overlap between s_j and u_i disappears.

Corollary 3. If $s_{j,u}$ moves across $u_{i,l}$ in negative direction with respect to the dimension, overlap between s_j and u_i disappears.

Corollary 4. If $s_{j,u}$ moves across $u_{i,l}$ in positive direction with respect to the dimension, an overlap between s_j and u_i emerges.

In this paper, the new algorithm for region matching is proposed based on these corollaries and Theorems given above.

4 Algorithm description

In this section, we show how our proposed DRBM algorithm derives the overlapping information rapidly and how it delivers region update efficiently with low network traffic.

4.1 Data structure

The data structure of our algorithm is shown in Figure 4. The array PsiSnaps keeps snapshots of update regions, and the array PhiSnaps keeps snapshots of subscription regions. The array PsiResiduals keeps

```
regionSnapshot[ ] PhiSnaps
regionSnapshot[ ] PsiSnaps
regionResidual[ ] PhiResiduals
regionResidual[ ] PsiResiduals
regionSet[ ] phiVector
regionSet[ ] psiVector
for each dimension
  boundList phiLowerBounds
  boundList phiUpperBounds
  boundList psiLowerBounds
  boundList psiUpperBounds
  int maxSRS
  int maxURS
  int DDT
```

Figure 4 Data structure.

residuals of update regions, and the array PhiResiduals keeps residuals of subscription regions (more details of snapshot and residual are provided in Subsection 4.3). The phiVector and psiVector are arrays which independently maintain the respective sets ϕ_i and ψ_j for each update and subscription region u_i and s_j respectively. For each dimension, there are four sorted bounds lists and two integers. All lists are sorted in ascending order. The phiLowerBounds keeps lower bounds of subscription ranges. The phiUpperBounds keeps upper bounds of subscription ranges. The psiLowerBounds keeps lower bounds of update ranges. The psiUpperBounds keeps upper bounds of update ranges. The integer maxURS and maxSRS hold the maximum update and subscription range sizes, respectively. The integer DDT is the dimension displacement threshold (more details in Subsection 4.3) in this dimension.

4.2 Overlap computation

For ease of explanation, we elaborate on our matching approach for an update region. An analogous description obtains for each subscription region.

Changes in each region are divided into two kinds according to its effect on overlap: *create a new region* and *modify an existing region*. A matching process is triggered by creating a new region to derive new overlapping information. A rematching process is triggered by modifying of a region to update the existing overlapping information.

A simple scenario with one update region u_i and five subscription regions located in a two-dimensional shared space is examined in Figure 5. The goal of our approach is to search overlapping information of the update region u_i with the least number of comparisons.

To simplify the searching process, lower bounds and upper bounds of all subscription regions are separately sorted in ascending order per dimension. Thus, all lower bounds and upper bounds of subscription ranges are first separately sorted in each common dimension [17]. For the scenario given in Figure 5(a), four lists are constructed:

1) List of lower bounds in x-dimension: x-phiLowerBounds = $\{s_{1,l}, s_{2,l}, s_{4,l}, s_{5,l}, s_{3,l}\}$;

2) List of upper bounds in x-dimension: x-phiUpperBounds = $\{s_{1,u}, s_{2,u}, s_{5,u}, s_{4,u}, s_{3,u}\}$;

3) List of lower bounds in y-dimension: y-phiLowerBounds = $\{s_{4,l}, s_{2,l}, s_{3,l}, s_{1,l}, s_{5,l}\}$;

4) List of upper bounds in y-dimension: y-phiUpperBounds = $\{s_{4,u}, s_{2,u}, s_{3,u}, s_{5,u}, s_{1,u}\}$.

After sorting bounds of regions in each dimension, the proposed DRBM approach handles dynamics of regions in accordance with the following.

Create a new region. When a new region is created, matching is performed with the following steps.

Step 1. For each dimension d, scan through the sorted lower bounds phiLowerBounds to get a set of regions with their lower bounds in the range ($u_{i,l}-$maxSRS, $u_{i,u}$) (According to Theorem 1).

Take the scenario shown in Figure 5(a) as an example, when an update region u_i is created, two sets are constructed: 1) $\text{tempSet}_x = \{s_2, s_4, s_5, s_3\}$ in dimension X, 2) $\text{tempSet}_y = \{s_4, s_2, s_3\}$ in dimension Y.

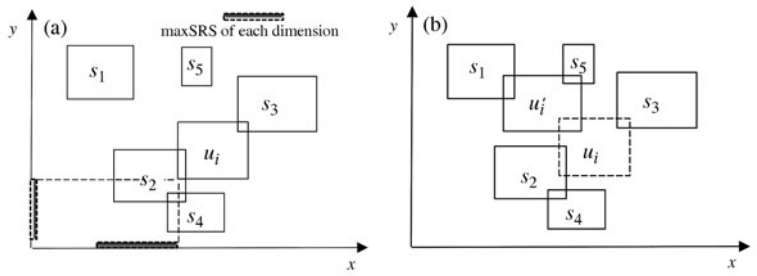

Figure 5 Basic scenario for the DRBM algorithm. (a) Matching triggered by creating a new region; (b) rematching triggered by region modification.

Step 2. For each result from Step 1, eliminate regions with upper bounds less than $u_{i,l}$.

In the example considered in Figure 5(a), two sets are constructed: 1) set $\phi_i^x = \{s_4, s_2, s_5, s_3\}$ which consists of regions overlap u_i in dimension X, and 2) set $\phi_i^y = \{s_2, s_3\}$ which consists regions overlapping u_i in dimension Y. With Inequalities 1, 2 and Theorem 1, both ϕ_i^x and ϕ_i^y are exact sets of ranges that overlap u_i in their respective dimension.

Step 3. Find the intersections of all sets derived from Step 2, and this is precisely the set ϕ_i, as in Eq. (3),

$$\phi_i = \bigcap_d \phi_i^d. \tag{3}$$

For the given scenario, we then have $\phi_i = \phi_i^x \cap \phi_i^y = \{s_4, s_2, s_5, s_3\} \cap \{s_2, s_3\} = \{s_2, s_3\}$.

Modify a region. If an update region u_i is modified, the rematching process is triggered to update the overlapping information. Rematching is performed with the following steps.

Step 1. For each dimension d, scan through the sorted upper bounds list phiUpperBounds to get a temporary set of regions with their upper bounds in the path of the moving lower bound, and scan through the sorted lower bounds list phiLowerBounds to get a temporary set of regions with their lower bounds in the path of the moving upper bound.

In the example considered in Figure 5(b), when u_i is updated to u_i' four temporary region sets are constructed: $\text{tempSet}_{x,l}$, $\text{tempSet}_{x,u}$, $\text{tempSet}_{y,u}$, $\text{tempSet}_{y,l}$, and $\text{tempSet}_{y,u}$. Therein, we have $\text{tempSet}_{x,l} = \{s_3\}$ by scanning through the x-phiLowerBounds in the range $(u'_{i,u}, u_{i,u})$, $\text{tempSet}_{x,u} = \{s_1\}$ by scanning through the x-phiUpperBounds in $(u'_{i,l}, u_{i,l})$, $\text{tempSet}_{y,l} = \{s_1, s_5\}$ by scanning through y-phiLowerBounds in$(u_{i,u}, u'_{i,u})$, and $\text{tempSet}_{y,u} = \{s_2\}$ by scanning through y-phiUpperBounds in $(u_{i,l}, u'_{i,l})$.

Step 2. For each dimension d, two sets are computed from the results of Step 1: 1) set ϕ_i^{d+} consisting of regions become overlap u_i, and 2) set ϕ_i^{d-} consisting of regions become separate from u_i.

Both sets ϕ_i^{d+} and ϕ_i^{d-} are computed according to corollaries given in Section 3. In the example considered in Figure 5(b), where $u_{i,l}$ moves in the negative direction of dimension X, we have with Corollary 1: $\text{tempSet}_{x,u} \in \phi_i^{d+}$. Similarly, with $u_{i,u}$ moving in the negative direction of dimension X, we can have with Corollary 3: $\text{tempSet}_{x,l} \in \phi_i^{d-}$; Also with $u_{i,l}$ moving in the positive direction of dimension Y, we have with Corollary 2: $\text{tempSet}_{y,u} \in \phi_i^{d-}$; And with $u_{i,u}$ moves in the positive direction of dimension Y, we can have with Corollary 4: $\text{tempSet}_{y,l} \in \phi_i^{d+}$. Finally, we have

$$\begin{aligned}
\phi_i^{x+} &= \emptyset \ \cup \ \text{tempSet}_{x,u} = \emptyset \cup \{s_1\} = \{s_1\};\\
\phi_i^{x-} &= \emptyset \ \cup \ \text{tempSet}_{x,l} = \emptyset \cup \{s_3\} = \{s_3\};\\
\phi_i^{y+} &= \emptyset \ \cup \ \text{tempSet}_{y,l} = \emptyset \cup \{s_1\} = \{s_1, s_5\};\\
\phi_i^{y-} &= \emptyset \ \cup \ \text{tempSet}_{y,u} = \emptyset \cup \{s_2\} = \{s_2\}.
\end{aligned}$$

Step 3. Compute $\phi_i^+ = \bigcup_d \phi_i^{d+}$ and $\phi_i^- = \bigcup_d \phi_i^{d-}$. Then for each dimension, remove items in ϕ_i^+ that do not overlap u_i by checking both inequalities 1 and 2.

For the given scenario, we have $\phi_i^+ = \{s_1, s_5\}$ and $\phi_i^- = \{s_2, s_3\}$.

Table 1 The matching algorithm

```
01. if(create a new region)
02.   if(an update region U_i is created)
03.     for each dimension d
04.       insert u_{i,l} into the psiLowerBounds
05.       insert u_{i,u} into the psiUpperBounds
06.       if (size of u_i > maxURS)
07.         maxURS = size of u_i
08.       end
09.       compute φ_i^d
10.     end
11.     compute φ_i as in Eq. (3)
12.     for each subscription region S_j ∈ φ_i
13.       add U_i to ψ_j
14.     end
15.   else //a subscription region is created
16.     similar to above ...
17.   end
18. end
19. if(modify an existing region)
20.   if(an subscription region S_i is modified)
21.     for each modified range s_i
22.       //s_i is in dimension d
23.       resort the phiLowerBounds of d
24.       resort the phiUpperBounds of d
25.       if (size of s_i > maxSRS)
26.         maxSRS = size of s_i
27.       end
28.       compute ψ_i^{d+} and ψ_i^{d−}
29.     end//end of for
30.     compute ψ_i^+ and ψ_i^−
31.     update ψ_i as in Eq. (4)
32.     for each update region U_j ∈ ψ_i^+
33.       add S_i to φ_j
34.     end
35.     for each update region U_k ∈ ψ_i^−
36.       remove S_i from φ_k
37.     end
38.   else //a update region is modified
39.     similar to above ...
40.   end
41. end
```

It is noteworthy that if $u'_{i,l}$ is greater than $u_{i,u}$ + maxSRS or $u'_{i,u}$ is smaller than $u_{i,l}$ − maxSRS in any dimension, all previous overlaps disappear according to both Theorems 1 and 2. Hence, $\phi_i^- = \phi_i$.We compute ϕ_i^{d+} by checking both inequalities 1 and 2 in $(u'_{i,l} - \text{maxSRS}, u'_{i,u})$ in this dimension, similar to creating a new range, instead of Steps 1 and 2.

Step 4. Update ϕ_i as in Eq. (4).

$$\phi_i = \phi_i \cup \phi_i^+ - \phi_i^-. \tag{4}$$

Hence, for the given scenario we have $\phi_i = \phi_i \cup \phi_i^+ - \phi_i^- = \{s_2, s_3\} \cup \{s_1, s_5\} - \{s_2, s_3\} = \{s_1, s_5\}$.

The main DRBM matching process is outlined in Table 1 using the notation and descriptors introduced before. A similar approach is applied to the subscription regions.

4.3 Region delivery

In DDM implementations, region delivery is an important aspect as bandwidth is wasted when all region data are transmitted even if only one range is changed. To prevent such waste caused by such region delivery, a region snapshot and a region residual are introduced in our DRBM strategy. A snapshot of a region is a whole copy of its data. For each region, there is only a unique version of a snapshot at any given simulation time. If a region snapshot is updated, its version should be increased accordingly. A region residual refers to modifications of a certain version of a snapshot.

To illustrate the basic idea of our snapshot and residual schemes, we first present an example. Figure 6 shows changes of a region R and its range in dimension X changes as the simulation proceeds. For simplicity, let region R only change in dimension X. As is shown in Figure 6(a), an initial version of region R's snapshot is first created in t_0. When R is modified, a residual is created by computing displacement of each range in all dimensions relative to the current snapshot. Take the scenario in Figure 6 as an example, when R is modified in t_1, only its range r_x is modified, thus a residual $r_x(a-c, b-d)$ is created. Then, only the t_1 residual $r_x(a-c, b-d)$ needs to be transmitted along with the version of the snapshot, then network load of region transmission is clearly lightened as a consequence. The receiver update the

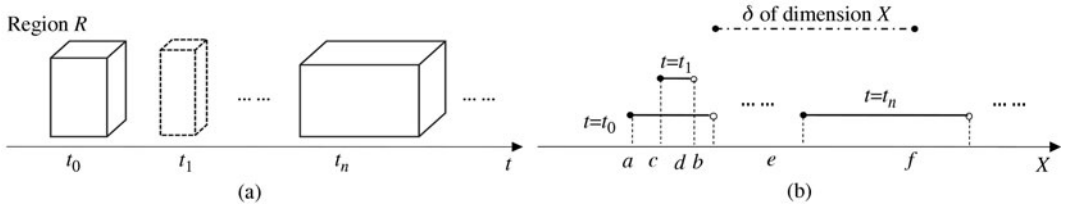

Figure 6 Snapshot and its residual of region R as the simulation time proceeds. (a) Region R changes as simulation time goes on; (b) range changes accordingly in dimension X.

relevant bound lists (i.e. phiLowerBounds, phiUpperBounds, psiLowerBounds and psiUpperBounds) by adding the residual to current snapshot of region R. In this way, the region update in DRBM could be a type of snapshot update packet, and a series of relative update packets with residuals referring to the current snapshot.

Furthermore, the network traffic can also be reduced by packaging, compressing, and then transmitting all existing residuals of all modified region each time. It is worth noting that the smaller the residual value, the higher compression ratio we can achieve, because of the zero value in the higher part. Hence, to keep residual value small, a factor called the dimension displacement threshold (δ) is introduced. Each dimension of the routing space has its own δ, and if displacement of a range node is greater than δ, a higher version of snapshot would be created by adding residual onto the current snapshot. Take the scenario in Figure 6 as an example, the displacement of upper node of r_x is $|b - f|$. A new version of snapshot should be created in t_n because $|b - f|$ is greater than δ.

5 Theoretical performance analysis

To obtain a better idea of the performance of the new algorithm, a theoretical performance is given here. The major parameters used in the algorithm's complexity analysis are given first:

- Number of update regions: N.
- Number of subscription regions: N (for simplicity, assumed to be the same as the number of update regions).
- Number of dimensions: D.
- Each dimension extends over $[0, L]$.
- Maximum range size of all regions: maxRS.

To analyze the storage complexity of our proposed algorithm, Figure 4 is reviewed. Both the PsiSnaps and PhiSnaps require $O(N)$ storage, as do PsiResiduals and PsiResiduals. According to [16], the average number of overlapping regions for a given region is $O((\text{maxRS}/L)^D * N)$. Hence, both phiVector and psiVector require $O((\text{maxRS}/L)^D * N^2)$ storage. The total storage for bound lists is $O(N)$. Both the maxSRS and maxURS require $O(1)$ storage. Overall, the total storage complexity of our DRBM algorithm is $O((\text{maxRS}/L)^D * N^2)$. Although it is quadratic storage complexity with respect to N, the actual storage requirement depends on the ratio of $O((\text{maxRS}/L)^D)$ which should be very small in a large spatial environment (e.g., 0.001^2).

To analyze the computational complexity of our algorithm, the matching algorithm in Table 1 is examined. The process of inserting bounds of a range into the respective bound list and calculating the maximum range sizes (Steps 04–08) requires $O(\log N + 1)$ computation. The process of computing ϕ_i^d (Step 09) requires $O(\text{maxRS}/L * N)$ computation. Step 11 requires $O(\text{maxRS}/L * N)$ computation. The computational complexity of Steps 12–14 is proportional to the average number of overlapping subscription regions for an update region, which is $O((\text{maxRS}/L)^D * N)$. If an existing region is modified, the process of resorting the respective bound list and calculating the maximum range sizes (Steps 23–27) requires $O(\log N + 1)$ computation. Step 28 requires $O(\text{maxRS}/L * N)$ computation. The computational complexity of Steps 30–37 is proportional to the average size of ϕ_i^{d+} and ϕ_i^{d-}, which is $O(\text{maxRS}/L * N)$.

In total, the computational complexity of our proposed algorithm in creating a new region or modifying an existing region is $O(\text{maxRS}/L * N)$ which is linear. With a small maxRS/L ratio (e.g., 0.001) in a

large spatial environment, the proposed DRBM algorithm could have high efficiency.

6 Experiment evaluation

In this section, the performance of the new algorithm is evaluated, including the computational performance of matching, the bandwidth occupied by region transmission, and the DDM services response delay.

To verify the correctness of theoretical conclusions made in Section 5, three C++ programs have been written to independently execute the region-based, Pan's improved sort-based, and DRBM algorithms. Each program performs a dynamic matching of a region modification and calculates the average performance for 50 iterations. All programs were run on a single PC with specifications Pentium D 3.40 GHz CPU and 1GB RAM on Microsoft Windows XP.

To show the performance comparison more clearly in different scale, two experiments are implemented on the basis of the number of regions. We set L=1000000 and maxRS=100. About 80% of the range displacement distances are controlled under maxRS*2 and the others are within a range of [maxRS*2, maxRS*10].

Figure 7 shows the total execution time of each algorithm for dynamic matching performance of one region modification and the total number of regions is no more than 30000. As shown in Figure 7, with the increase in the number of regions, the execution time of the region-based algorithm increases significantly and the other two algorithms increases slowly. The results show that our proposed matching algorithm has better computational performance as the number of regions increased.

Figure 8 shows the total execution time of each algorithm for dynamic matching performance of one region modification with a large number of regions. The results show that our proposed matching algorithm has a higher computational performance compared with that of the improved sort-based algorithm.

The performance of the region-based algorithm depends on the number of ranges, therefore another experiment was performed to establish the dependency of region size for the improved sort-based and our DRBM algorithm. We let the maxRS extend from 500 to 5000 in incremental steps of 500. The results in Figure 9 show that our proposed algorithm has better performance as the $\text{maxRS}/L * N$ ratio increased.

The storage required is an important factor for each algorithm, especially for a large scale environment. Hence, an experiment was devised with respect to a large number of ranges. The results in Figure 10 show that the improved sort-based and our proposed DRBM algorithms require more memory than that of the region-based algorithm, because both set up sorted bounds. As shown in Figure 10, a scenario with N=300000 DRBM algorithm takes less than 120 MB memory, which is acceptable.

We have implemented the proposed DRBM algorithm in BH RTI 2.3. To show that our proposed region delivery mechanism takes less bandwidth, a simple HLA federate application is designed and implemented the federate application on BH RTI 2.3, DMSO RTI 1.3NGv6 (non-bundle) and MÄK RTI 3.0. The experiment was performed with the number of federates extending from 2 to 8 in incremental steps of 1. Because the bandwidth occupied by region transmission is examined, each federate create 500 regions and call DDM services only. All federates were run on a single PC in the same local area network.

Figure 11 shows the bandwidth measured by the performance counter PerfCounter. The results show that BH RTI 2.3 takes much less bandwidth than DMSO RTI and MÄK RTI. It reduces the proportion of bandwidth occupied by about 15 percent.

A DDM service response delay refers to the time delay between the call of a DDM service from a caller and the instant an affected federate receives the result. To simplify the experiment, the DDM service interface registerObjectInstanceWithRegion is examined here. Because the experiment is conducted over a local area network, we make the assumption that the network delay is not taken into consideration. To verify the DDM services response delay with respect to range size, an experiment was performed with three federates. Each federate registered 10 subscription regions and 500 update regions and all regions are the same size. Results in Figure 12 show that each RTI performs stably. DMSO RTI takes a little more time because of the region-based matching algorithm.

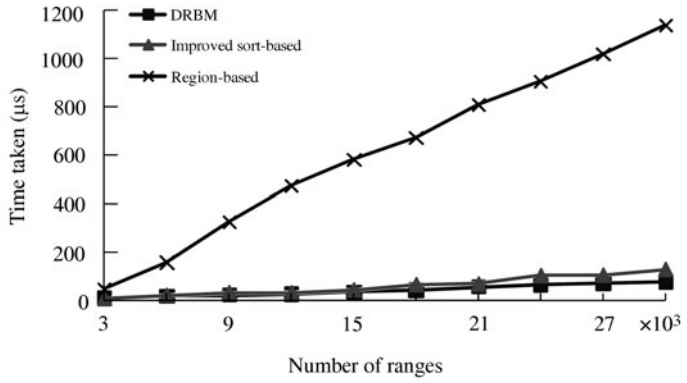

Figure 7 Performance with respect to number of ranges.

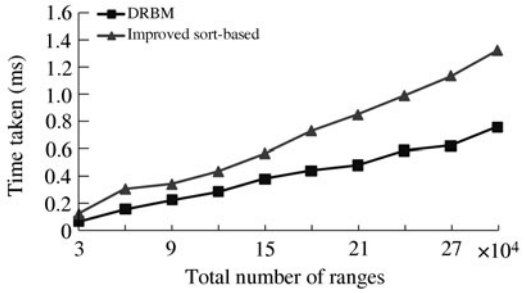

Figure 8 Performance with respect to a large number of ranges.

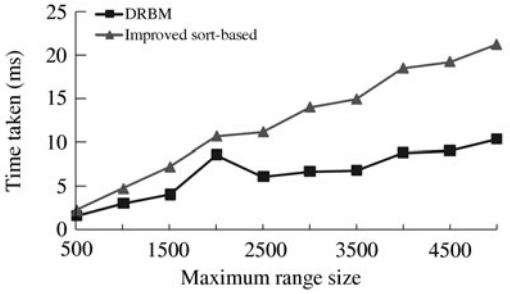

Figure 9 Performance with respect to maximum range size.

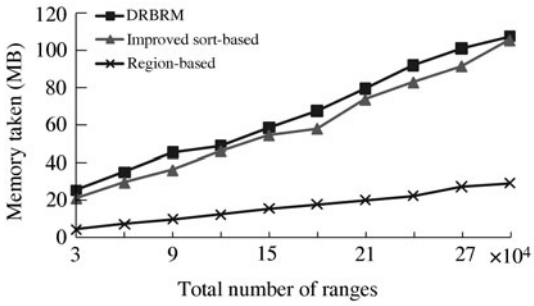

Figure 10 Memory taken.

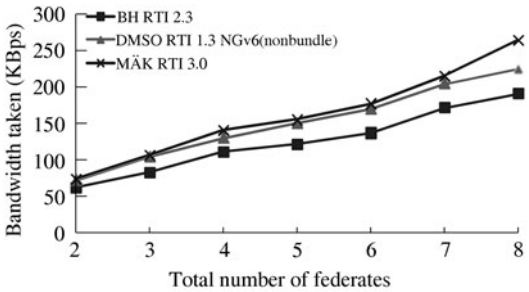

Figure 11 Bandwidth cost.

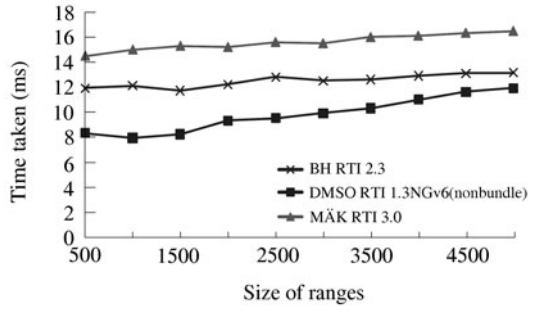

Figure 12 DDM services response delay with respect to range size.

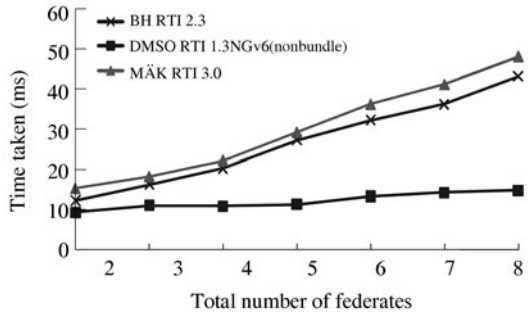

Figure 13 DDM services response delay with respect to the number of rederates.

To verify the DDM services response delay with respect to the number of regions, another similar experiment was performed with all range sizes set at 500. With the number of federates extending from two to eight, the total number of regions increase, and as shown in Figure 13, BH RTI 2.3 has a quicker

response time than DMSO RTI and MÄK RTI. Because both DMSO RTI and MÄK RTI are single-server architecture, server load increases rapidly under this condition. BH RTI 2.3 is in contrast a multi-server architecture; the server load could then be reduced.

7 Conclusions

In this paper, we propose a new DDM algorithm named DRBM which has good matching performance and significantly reduces network traffic caused by region transmitting.

Different from former works, we took both matching performance and network load caused by region transmission of DDM services into account. We first investigated the relationship between the region changes and overlap changes that can help to reduce matching and consequently improve matching efficiency. We classified region updates into two types of data expression, snapshot and residual. The network traffic could then be reduced by transmitting residual data instead of full region representations. Accordingly, the DRBM algorithm was devised. Experimental results show that DRBM has good matching performance and significantly reduces network traffic when the number of changing regions is large; therefore, it will play an effective role in simulations that are constrained by the network traffic.

Acknowledgements

This work was supported by National Basic Research Program of China (Grant No. 2009CB320805), Natural Science Foundation of China (Grant No. 61073070), 2008 China Next Generation Internet Application Demonstration sub-Project (Grant No. CNGI2008-123), and Fundamental Research Funds for the Central Universities of China.

References

1 IEEE. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Framework and Rules. New York: The Institute of Electrical and Electronics Engineer, 2000
2 IEEE. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Federate Interface Specification. New York: The Institute of Electrical and Electronics Engineer, 2000
3 IEEE. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Object Model Template (OMT) Specification. New York: The Institute of Electrical and Electronics Engineer, 2000
4 US Department of Defense. High Level Architecture (HLA) - Federate Interface Specification, Version 1.3. 1998
5 Van Hook D J, Calvin J O. Data distribution management in RTI 1.3. In: Proceedings of the 1998 Spring Simulation Interoperability Workshop. 1998
6 Ayani R, Moradi F, Tan G. Optimizing cellsize in grid-based DDM. In: Proceedings of the 14th Workshop on Parallel and Distributed Simulation. Washington: IEEE Computer Society, 2000. 93–100
7 Rak S J, Van Hook D J. Evaluation of grid-based relevance filtering for multicast group assignment. In: Proceedings of the Distributed Interactive Simulation, Orlando, 1996. 739–747
8 Lo S H, Chiu C A, Pai F P, et al. MGRID: a modifiable-grid region matching approach for DDM in the HLA RTI. In: Proceedings of the 2009 Spring Simulation Multiconference. San Diego: Society for Computer Simulation International, 2009
9 Abrams H, Watson K, Zyda M. Three-tiered interest management for large-scale virtual environments. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology. New York: ACM, 1998. 125–129
10 Minson R, Theodoropoulos G. An adaptive interest management scheme for distributed virtual environments. In: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation. Washington: IEEE Computer Society, 2005. 273–281
11 Morse K, Steinman J. Data distribution management in the HLA: multidimensional regions and physically correct filtering. In: Proceeding of the 1997 Spring Simulation Interoperability Workshop. Springer, 1997. 343–352
12 Shirmohammadi S, Kazem I, Ahmed D T, et al. A visibility-driven approach for zone management in simulations. Simul, 2008, 84: 215–229
13 Tan G, Zhang Y, Ayani R. A hybrid approach to data distribution management. In: Proceedings of the 4th IEEE International Workshop on Distributed Simulation and Real-Time Applications. Washington: IEEE Computer Society, 2000. 55–61

14 Zhang G, Zhang X, Li D. A hybrid DDM algorithm based on weight function. In: The 5th International Conference on Fuzzy Systems and Knowledge Discovery. Washington: IEEE Computer Society, 2008. 225–229
15 Pan K, Turner S, Cai W, et al. Implementation of data distribution management services in a service oriented HLA RTI. In: Proceedings of the 2009 Winter Simulation Conference. Austin: Winter Simulation Conference, 2009. 1027–1038
16 Pan K, Turner S, Cai W, et al. An efficient sort-based DDM matching algorithm for HLA applications with a large spatial environment. In: 21st International Workshop on Principles of Advanced and Distributed Simulation (PADS 07). Washington: IEEE Computer Society, 2007. 70–82
17 Raczy C, Tan G, Yu J. A sort-based DMM matching algorithm for HLA. ACM Trans Model Comput Simul, 2005, 15: 14–38
18 Capps M, Stotts D. Research issues in developing networked virtual realities. In: Proceedings of the 6th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). Washington: IEEE Computer Society, 1997. 205–211
19 Macedonia M, Zyda M, Pratt D, et al. Exploiting reality with multicast groups: a network architecture for large-scale virtual environments. IEEE Comput Graph, 1995, 15: 38–45
20 Sorroche J, Szulinski J. Bandwidth reduction techniques used in DIS exercises. In: Proceedings of the 2004 European Simulation Interoperability Workshop. Orlando: SISO's Digital Library, 2004
21 Torpey M, Wilbert D, Helfinstine B, et al. Experiences and lessons learned using RTI-NG in a large-scale, platform-level federation. In: Simulation Interoperability Workshop. Orlando: SISO's Digital Library, 2001