Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
88(11) 1350–1367

DOI: 10.1177/0037549712450628
sim.sagepub.com
SAGE SCS

A binary partition-based matching algorithm for Data Distribution Management in a High-level Architecture-based distributed simulation

Junghyun Ahn, Changho Sung and Tag Gon Kim

Abstract

Data Distribution Management (DDM) is one of the High-level Architecture (HLA) services that reduce message traffic over the network. The major purpose of DDM is to filter and route the exchange of data between federates during a federation. However, this traffic reduction usually results in a significant computational overhead, which is caused by calculating the intersection between update regions and subscription regions in a matching process. To reduce the computational overhead for the matching process, this paper proposes a binary partition-based matching algorithm for DDM in a HLA-based distributed simulation. The new matching algorithm is fundamentally based on a divide-and-conquer approach. The proposed algorithm recursively performs binary partitioning that divides the regions into two partitions that entirely cover those regions. This approach promises low computational overhead, since it does not require unnecessary comparisons within regions in different partitions. The experimental results show that the proposed algorithm performs the existing DDM-matching algorithms better and improves the scalability of the DDM.

Keywords

Data Distribution Management, High-level Architecture/Runtime Infrastructure, large-scale High-level Architecture-based distributed simulation, matching algorithm, binary partition, scalability

1. Introduction

Data Distribution Management (DDM) is one of the most important filtering mechanisms in large-scale distributed simulations.[1–3] DDM has been successful in reducing the network traffic in some respect, but its capability is limited by the computational overhead for matching update regions and subscription regions that represent the interests of data producers and data consumers, respectively.[4] For example, a typical DDM scenario is a battlefield simulation, where sensors are deployed to detect enemy movement. The enemy units are moving to a certain destination point and a sensor detects any enemy units. The update regions represent the location of enemy units with small rectangles, and the subscription regions represent the detecting ranges of the sensors. In this example, only if the update regions of the enemy units and subscription regions of the sensor overlap, the location of the enemy units will be routed by the DDM filtering mechanism. However, that mechanism produces high computational overhead because a matching process is performed to calculate the intersection between all pairs of update regions and subscription regions.

Several studies of the DDM-matching algorithms have been proposed in recent years.[5–8] For instance, a region-based algorithm[9] exhaustively compares the intersection of all the pairs of regions with high computational overhead, but it achieves exact matching. On the other hand, the grid-based algorithm[6] separates a multidimensional space into a regular grid. In place of exhaustive matching computing, update regions and subscription regions are assumed to overlap if and only if they share at least one common grid cell. Therefore, the grid-based algorithm

Department of EE, KAIST, Republic of Korea

Corresponding author:
Junghyun Ahn, Department of EE, KAIST, 335 Gwahangno, Yuseong-gu, Daejeon, 305-701, Korea.
Email: jhahn@smslab.kaist.ac.kr

provides much less computation of the matching process than the region-based algorithm, but it sacrifices accuracy. While all of the DDM-matching algorithms have been devised to balance considerable computational overhead and the accuracy, they are not aware of how the regions are generated and distributed in the multidimensional space. As the previous matching algorithms do not consider the characteristics of the region distribution, it is difficult to select a matching algorithm that is appropriate for some region-distribution situations. In the case of large-scale distributed simulations, such as a battlefield simulation, it will probably be a situation with large overlapping regions. The computational overhead of the matching process in large-scale simulations is significant, because of superfluous intersections between regions.

Therefore, in this paper, we propose a binary partition-based matching algorithm based on the divide-and-conquer approach to reduce the computational overhead in the matching process for the DDM. The design goal of our algorithm is to perform binary partitioning that divides the regions into two partitions, the *left-hand* partition and *right-hand* partition, and to find out the overlap information that entirely cover those regions recursively, and then carry out the matching process that calculates the intersection of regions in the partition boundary, since these regions in the partition boundary can be included within two partitions. We assume that the regions on the partition boundary are located in the left-hand partition and define these regions as a *pivot set*, which is illustrated in Section 4.1. Specifically, the matching process is executed by comparing the intersections between regions that correspond to boundaries in the pivot set of the left-hand partition and regions in the right-hand partition.

Figure 1 shows a conceptual overview of the binary partitioning for the matching process. Typically, we assume that the multidimensional space is a two-dimensional space in Figure 1. The algorithm first performs the *x*-dimension at a partitioning run and then repeats this procedure for the *y*-dimension.

As shown in Figure 1(b), in the binary partitioning on the x-dimension, the update or subscription regions tend not to fall on the boundaries of partitions, because the region size is often smaller than the partition size. To reflect the region distribution, the proposed algorithm uses the concept of an ordered relation that represents the relative location of the partition in the multidimensional space. If an update region in the left-hand partition and a subscription region in the right-hand partition exist, there is no need to calculate the intersection between these regions, which leads to the reduction of the matching operation that is used in the matching process, because these regions exist in different partitions that are located in the ordered relation of partitions. The previous matching algorithms we have examined so far do not focus on the relative location of partitions to improve the overall performance of the matching process.

In the matching process, the proposed algorithm has the ability to classify whether region boundaries fall accurately on partition boundaries. As illustrated in Figure 1(b), the regions in the partition boundary will all overlap in the *x*-dimension. Thus, the proposed algorithm can easily find the calculation of the intersection of the *x*–*y* dimensions in the matching process. This requires minimal computational overhead in some partitions, in particular those with large overlapping regions or in a clustered distribution in which regions are collected at particular points. In addition, as the

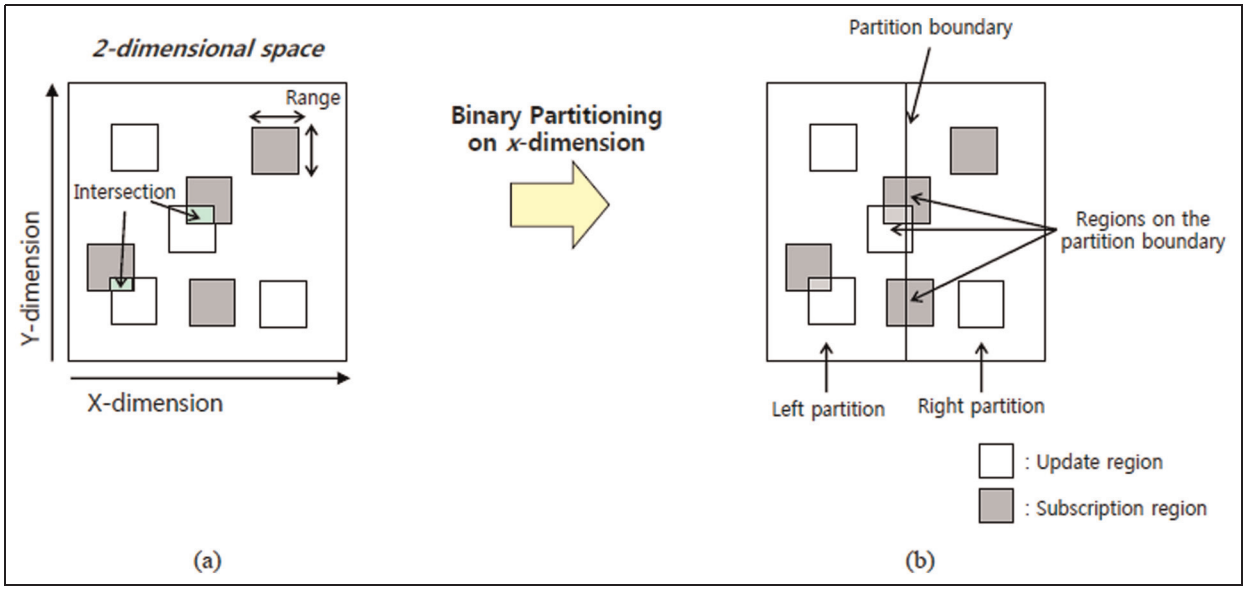

Figure 1. A conceptual overview of the binary partition-based matching algorithm.

proposed algorithm considers the ordered relation of partitions to reflect the characteristics of the region distribution over a range of overlapping conditions, it provides robustness in terms of a dynamic range measure on variations in the overlapping regions.

To evaluate the proposed algorithm, a set of experiments has been conducted with different numbers of regions, overlap rates, and distribution of regions for various situations to reflect the region distribution. Our experimental results show that the proposed algorithm significantly performs better than the previous DDM-matching algorithms in the execution time for the matching process, as well as in the dynamic range measure. The proposed algorithm outperforms the previous matching algorithms by more than 45%, on average, in terms of execution time, when many regions are greatly overlapped. The more regions are overlapped, the better the performance of the proposed algorithm will be. In more complex scenarios where the distribution of regions is clustered, not uniform, the proposed algorithm is much better than the others and improves the scalability of DDM implementation for large-scale simulations. Moreover, the proposed algorithm is more robust to variations in the overlap rate than the others for the different characteristics of the region distribution in terms of the dynamic range measure.

The remainder of this paper is organized as follows. Section 2 describes an overview of the DDM in the High-level Architecture (HLA), and Section 3 briefly outlines the work related to this paper. In Section 4, we present a binary partition-based matching algorithm based on a divide-and-conquer strategy. Section 5 shows experimental results across a variety of workloads. Finally, Section 6 gives concluding remarks.

2. Overview of Data Distribution Management in High-level Architecture

As a standard architecture for interoperation between heterogeneously distributed simulations that are developed by different languages and platforms, the HLA was approved as the Institute of Electrical and Electronics Engineers (IEEE) Standard 1516 in 2000.[10–13] A revised version, HLA Evolved, has now also been standardized by IEEE 1516-2010.[14–16] It provides a common framework within which simulation developers can structure and describe their simulations. The individual simulation model is referred to as a federate. A federation is defined as the grouping of federates to achieve some specific objective of simulation interoperation. The HLA standard formally consists of three main parts: (1) HLA Rules,[14] which are a set of 10 basic rules that define the interoperability and the responsibilities of federates and federations; (2) Federate Interface Specification,[15] which provides a specification that a simulation may perform, or be asked to perform, during HLA federation execution; it includes Runtime Infrastructure (RTI) services that are available to each simulation and the callback functions that each federate must provide to the RTI; and (3) the Object Model Template (OMT),[16] which is a common template for specifying simulation information in terms of a hierarchy of object classes, their attributes, and so on. It forms a documentation standard for a description of a data model.

The aim of HLA is to promote interoperability between simulations and to encourage the reuse of simulations in various contexts. HLA's interoperability can be achieved by conforming to the Federate Interface Specification. If the simulations are developed under the HLA standard, HLA establishes a common technical framework to facilitate the interoperability of simulations. HLA's reusability can also be achieved through the use of the OMT. The essential sharable simulation information in a particular federation is described by the Federation Object Model (FOM), a unified data model within the federation that is documented in accordance with the OMT. A FOM and an existing federate can easily be adapted and reused in another federation.

Because HLA is an architecture, not software, RTI software is required to support the IEEE 1516.1 Federate Interface Specification. RTI is an implementation of the services described in the Federate Interface Specification. Through a federate's Local RTI Component (LRC), the RTI provides each federate with management services, such as federation management, time management, object management, and DDM. In general, as a customized version, RTI software consists of three parts: the RTI executive process, federation executive process, and LRC library.[17]

The RTI is often used as the distributed operating system for large-scale HLA-based distributed simulations.[18–23] It makes possible simulations to construct distributed simulations that facilitate the execution of simulations across multiple nodes and improve simulation performance. In large-scale simulations, a simulation entity updates objects that need to be exchanged over distributed simulation nodes. These simulations have tens of thousands of simulation objects that need to be exchanged in real time for interoperation.[24,25] This may incur a large processing overhead on the simulation nodes and large communication overhead on the network.[26] Such an application may require a filtering mechanism to reduce communication traffic among the federate nodes over the network.

DDM services, one category of the HLA management services, reduce message traffic by sending the data only to those federates that need the data. The DDM performs filtering based on the interest expressions of federates in a federation and allows the federation to control the routing and delivery through user-defined information (i.e.

Table 1. Terminology definitions in Data Distribution Management (DDM).[14]

Terminology	Definition
Dimension	A named coordinate axis with non-negative integers
Multidimensional space	A coordinate system whose dimension is *d* (where d is a fixed natural number)
Range	A continuous semi-open interval on a dimension (lower bound, upper bound]
Region	A set of ranges for any given dimension
Update region	A specified set of region instances associated with a publishing federate
Subscription region	A specified set of region instances associated with a subscribing federate
Overlap	All ranges of dimensions that lie over and partly cover in the update region and subscription region put upon another pairwise
Intersection	A situation in which the corresponding region sets overlap
Matching process	A process to calculate the intersection between update regions and subscription regions

regions). Using the DDM, the RTI then distributes objects and interactions by sending federates to receiving federates. The DDM services allow value-based filtering.[1] This type of filtering provides the most precise filtering mechanisms, which ensures that the federates receive the minimal set of data they are interested in. In a large-scale federation, it is necessary to filter and route more elaborately for data exchange during the federation execution. Therefore, the DDM provides value-based filtering for large-scale simulations that both require long execution time and consume vast memory space.

Table 1 presents the definitions of terms used in this paper. These definitions originate from the HLA standard.[14] Although the definitions of terms are provided in Table 1, the terms' usage is illustrated and described in Figure 1(a). This figure shows an example of region intersection for value-based filtering. Because our interest lies in the simple exemplification of DDM terms, we assume that the multidimensional space is a two-dimensional space. In the figure, there are four update regions and four subscription regions. An update region is the defined set of data declared by a publishing federate, whose information is delivered to subscribing federates. A subscription region is the area of interest declared by the subscribing federates. A situation in which the corresponding region sets overlap is called an *intersection*. When an intersection exists, data exchange occurs from the publishing federate to the subscribing federate. In this case, there are two intersections between the update regions and subscription regions. The process of identifying these intersections between update regions and subscription regions is a *matching process*. The main role of DDM is to reduce the volume of data exchanged through the matching process during a federation.

With the subscription region and the update region, the publishing federate's LRC performs the matching process, which is the focus of this paper. It is necessary for a publishing federate's LRC to know exactly who is subscribing. The publishing federates want to send their updates to all joined federates, so it checks if it matches some of its subscription regions for DDM considerations (i.e. it must know who to notify of the subscription region changes). The objective of the matching process is to efficiently obtain the overlap information when update regions and subscription regions intersect. Since this process leads to higher computational overhead and to less data transmission amount on the network, the performance of DDM depends on how well the matching process is performed.

3. Related works

Previous matching algorithms have solved matching problems differently depending on their various performance concerns. Table 2 briefly summarizes the major characteristics of the previous matching algorithms. We compare the proposed method with the previous matching algorithms in Section 5. We have classified them into several categories as follows:

1) the region-based algorithm;[1,2]
2) the grid-based algorithm and the hybrid approach;[6,27]
3) the sort-based algorithm;[7]
4) the other DDM-matching algorithm.

Firstly, the region-based algorithm *exhaustively* compares the intersection of all the pairs of update regions and subscription regions. This algorithm is quite straightforward while computing the intersection, as every update region is checked directly against every subscription region. If there are N update regions and M subscription regions, there are $N * M$ pairs to check in the worst case. If there are many intersections in a large, spatial simulation because of a high number of simulated entities, a considerable amount of computational overhead occurs in the matching process. However, the exact overlap information between update regions and subscription regions can be found. In addition,

Table 2. Comparison of previous matching algorithms.

Matching algorithms	Description	Advantage	Disadvantage	Average case complexity
Region-based algorithm	Brute-force method	Simple implementation	Too time consuming	$d * O(N^2)$
Hybrid approach	Based on the grid-based algorithm	Exact match	Needed optimal grid size	$d * c * O(N^2)$
Sort-based algorithm	Sorting algorithm usage	On average performance	Needed to optimize the sorting data structure	$d * N * O(\log N)$
Proposed method	Binary partitioning	Scalability; robustness	Needed to optimize initialization costs in partitioning	$d * N * O(\log N)$

N = # of regions, d = # of dimensions, c = # of grid cell.
As first noted by Petty and Morse,[4] DDM Rectangle Matching (DRM) requires time with a lower bound in Ω (N* log N) and upper bound in $O(N^2)$, where N is the number of runtime data distribution actions performed by the federates.

the region-based algorithm is directly implemented in the matching operation in the DDM of the RTI.

The grid-based algorithm involves dividing the multidimensional space into a grid of cells, and update regions and subscription regions are allocated into each cell of the grid.[6,28–30] In this algorithm, as an exact evaluation of the matching process is not implemented, update regions and subscription regions are assumed to overlap if and only if they share at least one common grid cell. Although the grid-based algorithm requires much less computation than the region-based algorithm, incorrect matching creates irrelevant data communication, and additional receiver-side filtering is required. Another problem is that it is hard to define the appropriate size of the grid cells.[31] The larger grid cell size causes more irrelevant data to be received by each receiving federate. On the other hand, when the grid cells are too small, each region is mapped on several different cells, which leads to redundant computations.

On the other hand, the hybrid approach improves performance by exploiting the advantages of the two previous approaches (i.e. the region-based algorithm and the grid-based algorithm) and minimizing the drawbacks of the grid-based algorithm.[27,32] This approach uses a variation of the region-based approach, which uses grid cells to reduce the irrelevant communication overhead. It first minimizes the irrelevant message cost of grid-based filtering to map all regions to the grid cells. Then, the region-based approach is used to make the exact match by checking a pair of update regions and subscription regions within each cell. In this way, the computational overhead of the matching process is lower than that of the region-based algorithm, and the overlapping information is exact. It also produces a lower number of irrelevant messages than the grid-based algorithm. However, building an irregular grid may occur with the hybrid approach as well as with the grid-based algorithm. The major issue of this approach is selecting the optimal size of the grid cells, which the performance of the hybrid approach depends on.

There are several other matching algorithms that have been used in HLA/RTI, such as the sort-based algorithm detailed by Raczy et al.;[7] Pan et al.[33] compute the intersection between update regions and subscription regions using a sorting algorithm. The sort-based algorithm projects the multiple dimensions of regions on each dimension. The end points in each dimension of all regions are sorted for each direction to determine the overlap information. When the sorted lists of end points are scanned, the sort-based algorithm can maintain the set of subscription regions before and after the current position. Therefore, it is possible to know exactly, for each update region, which subscription regions match on each dimension, which have a time complexity of $N * O(\log N)$, where N is the total number of regions. However, the sort-based algorithm's performance is degraded when the regions are highly overlapped, and it is necessary to optimize the sorting data structure for an efficient matching operation.

In addition to these approaches, the off-loading matching algorithm transfers the matching process with another device.[8,34,35] There is a coarse-grained task parallelism between the matching process and the other DDM operations. When regions are added or modified by federates, only the information from updated regions is copied from the host to the device. In Santoro and Fujimoto,[8] a part of the matching computation is shifted to network processors or, in Lo,[35] to graphical processing units that are based on the Compute Unified Device Architecture (CUDA). An approach using agents for matching process was proposed by Tan et al.[5] and Wang et al.[36] An agent-based DDM approach has been proposed to filter user messages on the sender side to avoid transmitting unnecessary data to the subscribers by using the agents. In addition, a quadtree-based approach that is similar to the hybrid approach decomposes the multidimensional space into equal-sized quadrants, similar to the grid cells.[37] Instead of using grids to partition the space, this approach uses a quadtree for the decomposition process.

All of the previous works have been devised to reduce the considerable computational overhead. Since the

previous matching algorithms are not aware of how regions are generated and distributed in the multidimensional space, it is difficult to select a matching algorithm that is appropriate for some region-distribution situations. As described in Table 2, there are matching algorithms to apply for the different situations case by case. These algorithms are with a time complexity of a lower bound in $\Omega(N * \log N)$, where N is the number of runtime data distribution actions performed by the federates. However, they have a quadratic complexity $O(N^2)$ in the worst case.[4]

Hence, we present a binary partitioning algorithm to contribute to these existing works, which is somewhat more efficient and has the advantage of being very robust. In the proposed method, we focus on the ordered relation of partitions to reflect the region distribution. The proposed method takes advantage of reducing computational overhead and improves the overall performance of the matching process through the binary partitioning. The proposed method achieves scalability in the system across a variety of workloads. In addition, it provides robustness in terms of a dynamic range measure on variations in the overlapping regions. The binary partitioning approach, of course, has an initialization cost in the partitioning procedure. Hence, it is necessary to optimize an efficient algorithm and reduce overhead in terms of initialization costs.

4. Binary partition-based matching algorithm

4.1 Principle of the proposed algorithm

In this section, we present the design principle of the binary partition-based matching algorithm. Our approach takes a divide-and-conquer approach similar to the one used for the Quicksort algorithm,[38] which is roughly analogous to the use of pivots in sorting, with the potential to cut down the overall number of comparison operations. This approach consists of two main processes, the repetitive binary partitioning process and the matching process.

Firstly, in the binary partitioning process, the algorithm recursively performs binary partitioning, which divides the regions into two partitions that entirely cover those regions by a *pivot value*. The partitioning operation is started by selecting the pivot value and a dimension. The regions are partitioned into approximately two equal-sized sets, which are assigned to the left-hand and right-hand partitions. Regions are assigned to the right-hand partition if the coordinate (i.e. the regions range) in that dimension is greater than the pivot value. Otherwise, regions are assigned to the left-hand partition if the regions range in that dimension is less than or equal to the pivot value. In addition, regions for which the coordinate includes the pivot value in the regions' range are assigned to the left-hand partition.

If a region falls on the partition boundary, the region is assumed to be assigned to the pivot set in the left-hand partition. All regions' range in the pivot set of the left-hand partition includes the pivot value. This value can be a *midpoint*, which reduces the probability of choosing a bad pivot value. The midpoint guarantees equal-sized partitions with some computational costs for the binary partitioning, approximately. This is the optimal and most appropriate situation when the distribution of regions is not known. Therefore, in this paper, we assume that the pivot value is the midpoint in the regions. Although the selection of the pivot value as the midpoint is intuitive and faster, this selection of the midpoint may lead to imbalanced partitions when the distribution of regions is not uniform. Rather than an arbitrary midpoint as an approximation, any information on the statistical analysis can be used as a better approximation of the pivot value.[38,39] However, the midpoint is used as the representative value in many application domains because it does not consider the distribution of regions.

Secondly, in the matching process, the algorithm uses the concept of an ordered relation, which represents the relative location of the partition. After the binary partitioning, update or subscription regions tend not to fall on the boundaries of partitions. If an update region in the left-hand partition and a subscription region in the right-hand partition exist, there is no need to calculate the intersection between these regions, which leads to the reduction of a matching operation that is used in the matching process. Because these regions exist in different partitions that are located in the ordered relation of partitions, the matching operation is performed for the calculation of the intersection of regions in the partition boundary. In other words, this matching operation is executed by comparing the intersections between regions that correspond to boundaries in the pivot set of the left-hand partition and regions in the right-hand partition. This requires minimal computational overhead in some partitions, especially in large overlapping regions. The significant points of our algorithm are that these partitions are being processed in parallel. The proposed algorithm promises low computational overhead, since it easily calculates the intersection between regions on partition boundaries and does not require unnecessary comparisons within regions in different partitions. Therefore, in this paper, we propose a binary partition-based matching algorithm based on the divide-and-conquer approach to reduce the computational overhead in the matching process for the DDM. The design goal of our algorithm is to perform binary partitioning that divides the regions into two partitions, the *left-hand* partition and *right-hand* partition, and to find out the overlap information that entirely covers those regions, recursively.

4.2 Dimension projection

To facilitate a meaningful interpretation of the multidimensional space, our approach uses a well-known dimension projection algorithm.[7,33,34,40] Before presenting the dimension projection algorithm, we introduce our approach to a region representation as follows. Let a collection of the projected regions $\mathsf{T}\mathbb{R}^d = \{ Rj^d, r(i, j)^d \}$, where $i \in$ (up, sub) (i.e. i is a flag to show whether a region is up, update region, or sub, subscription region), a region identifier $j = 1,\ldots,n$, and d = a specified dimension. $(r(i, j)^d.l, r(i, j)^d.u)$ is the range of regions whose upper bound is $r(i, j)^{d\ d}.u$ and lower bound is $r(i, j)^d.l$ with the i the update/subscription flag and the j region handle on the d dimension. The dimension projection algorithm is used to determine the overlap information spatially in multidimensional space. Previous works on the regions in multidimensional space are not intuitive, so the dimension projection algorithm is used, defined as follows:

Definition 1. Dimension projection, T: the d-dimensional projection is $\mathsf{T}\mathbb{R}^d$, where $\mathbb{R}$ is the set of all regions and dimension $d \geq 1$.

The algorithm first works by projecting all dimensions in the multidimensional space, using the definition shown above. It reduces the multidimensional problem to a one-dimensional problem and intuitively finds the pairs of regions may overlap. It then carries out the partition process, as presented in Section 4.3. Figure 2 shows the dimension projection of eight regions located in a two-dimensional space. In the figure, we project each region to the x-dimension. Next, the x-dimension is recursively partitioned into a binary partition. After performing the procedure for one dimension, the algorithm repeats this procedure for the y-dimension. The overall overlap information can be obtained by combining the information of each dimension: two regions overlap if and only if they overlap for the x–y dimensions.

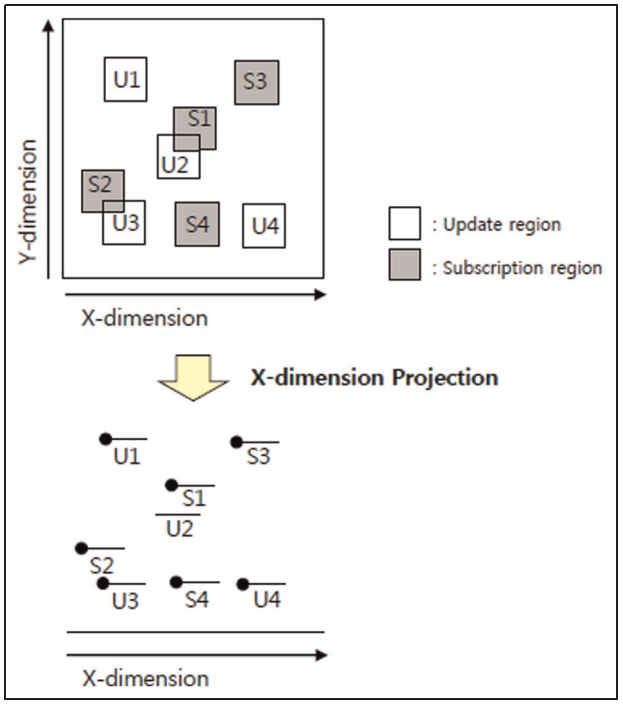

Figure 2. Dimension projection with the x-dimension.

4.3 Proposed binary partition-based matching algorithm

We present the overview of the binary partition-based matching algorithm. After regions **R** are projected by the d dimension, the original projected region set $\mathsf{T}\mathbb{R}^d$, our algorithm divides the projected regions $\mathsf{T}\mathbb{R}^d$, which are partitioned in approximately two equal-sized sets, which are assigned to the left-hand and right-hand partitions, P_l and P_r. This guarantees the avoidance of checking the intersection between regions in the different partitions. Regions are assigned to the right-hand partition if the coordinate (i.e. the regions range) in that dimension is greater than the pivot value. Otherwise, regions are assigned to the left-hand partition if the region range in that dimension is less than or equal to the pivot value. In addition, regions for which the coordinate includes the pivot value in the region range (i.e. pivot set) are assigned to the left-hand partition.

The conceptual description shown in Figure 3 demonstrates the binary partition. After the dimension projection, the algorithm uses the binary partitioning of the regions around a pivot value and then deals with the two smaller partitions separately. As a result, the x-dimension is partitioned into two partitions by the pivot value. They are the left-hand partition of the pivot value and the right-hand partition of the pivot value. Each partition consists of one or several regions. Any partitions are disjoint with respect to each other, which are made of different regions, as described in partition properties.

By using the binary partitioning, projected regions $\mathsf{T}\mathbb{R}^d$ are decomposed into several partitions, $\mathbb{P} = \{\mathcal{P}_1, \mathcal{P}_2 \ldots, \mathcal{P}_m\} \leftarrow$ such that:

1) $\forall p \in \{1, 2, \ldots, m\}, \forall j \in \{1, 2, \ldots, r\}, \forall p_r \in \{1, 2, \ldots, n\}, P_p = \{R_{p1}, R_{p2}, \ldots, R_{pr}\}$, each partition consists of a partition identifier (p) and belongs to $\mathbb{P}$;
2) 2) for $\exists R \in \mathsf{T}\mathbb{R}^d$ and $\exists p \in \{1, 2, \ldots, m\}$, R belongs in a partition, $R \in P_j$, since P_j belongs to $\mathsf{T}\mathbb{R}^d$;
3) $\forall p, q \in \{1, 2, \ldots, m\}$; if $p \neq q$, then $P_p \cap P_q = \varnothing$, and regions are distinct, since these regions are located in the different partition;
4) the union of P is all regions $\mathsf{T}\mathbb{R}^d$.

Table 3. Three partitions of the binary partition.

Partition	Set of update and subscription regions		Region
Pl	S_{lu}		{U3, U1}
	S_{ls}		{S2}
	Pivot set	S_{pu}	{U2}
		S_{ps}	{S1, S4}
Pr	S_{ru}		{U4}
	S_{rs}		{S3}

Through these *partition properties*, we ensure that the regions are correctly represented by the set of partitions. The partition of regions can be more than two with the projected region set. However, our interest is in the *two subsets* case by using the binary partition. Therefore, we provide the following definition of the binary partition.

Definition 2. The binary partition: after regions $\mathbb{R}$ are projected by the d dimension, the original projected region set $\mathsf{T}\mathbb{R}^d$ is divided into two partitions, $\mathcal{P}_l$ and $\mathcal{P}_r$, by choosing a pivot value as the boundary. A binary partition of $\mathsf{T}\mathbb{R}^d$ is a pair of region subsets $(\mathcal{P}_l, \mathcal{P}_r)$, such that $\mathsf{T}\mathbb{R}^d) = \mathcal{P}_l + \mathcal{P}_r$, $\mathcal{P}_l \mathcal{P}_r \leq \mathsf{T}\mathbb{R}^d$, and $\mathcal{P}_l \bigcup \mathcal{P}_r = \mathsf{T}\mathbb{R}^d$. For example, the binary partitioning of $\mathsf{T}\mathbb{R}^d$ results in $(\mathcal{P}_l, \mathcal{P}_r)$.

Algorithm 1. Binary partition algorithm.

Input: The projected Regions $\mathsf{T}\mathbb{R}^d$, pivot value p
Output: Sorted regions set, S_{lu}, S_{ru}, S_{pu}, S_{ls}, S_{rs}, S_{ps}
1: **procedure** BinaryPartition($\mathsf{T}\mathbb{R}^d$, p)
2: **for** Each region $R_i \in$ Region $\mathsf{T}\mathbb{R}^d$ **do**
3: **if** $\mathcal{R}j$ = *update region* **then**
4: **if** $r(u\ j)^d.l \leq p$ **then**
5: $S_{lu} \leftarrow \{j\} \bigcup S_{lu}$
{*pivot set in the left partition*}
6: **if** $r(u, j)^a.u > p$ **then**
7: $S_{pu} \leftarrow \{j\} \bigcup S_{pu}$
8: **end if**
9: **else if** $r(u, j)^d.l > p$ **then**
10: $S_{ru} \leftarrow \{j\} \bigcup S_{ru}$
11: **end if**
12: **else if** $\mathcal{R}j^d$ = *subscription region* **then**
13: {*similar to above*}
14: **end if**
15: **end for**
16: **return** S_{lu}, S_{ru}, S_{pu}, S_{ls}, S_{rs}, S_{ps};

The detailed binary partition algorithm is presented as Algorithm 1. Those regions in which the regions' ranges are greater than the pivot value are put in $\mathcal{P}_r$; otherwise, they are put in $\mathcal{P}_l$. This partition consists of four subsets of sorted regions, S_{lu}, S_{ls}, S_{pu}, and S_{ps}, in ascending order, that is, the update regions and subscription regions for the left-hand partition and the update regions and subscription regions for the pivot set in the left-hand partition, respectively. In addition, $\mathcal{P}_r$ consists of two subsets of sorted regions, S_{ru} and S_{rs}. For the matching process, S_{pu} and S_{ps} in the pivot set are compared and matched with the other set in the left-hand and right-hand partitions, S_{lu}, S_{ls}, S_{ru}, and S_{rs}. Table 3 presents three partitions of the binary partition. In Figure 3, from the left-hand partition $\mathcal{P}_l$, we find that *Slu* is {U3, U1} and S_{ls} is {S2}. The other set in each partition is also arranged by the binary partition, which is illustrated in Table 3.

Algorithm 2. Intersection calculation algorithm.

Input: Sorted regions S_l, S_r; dimension d
Output: n-by-n matrix, $OM^d = (omij^d)$, where $i, j \in n$
1: **procedure** IntersectionCalculation(S_l, S_r)
2: *Smatching* = $\varnothing$,
3: $i = 0$;
4: $j = 0$;
5: **for** *Each update region* $R_i^d \in S_l$ and *each subscription region* $R_j^d \in Sr$ **do**
6: **if** $r(u, i)^d.u > r(s, j)^d.u$ **then**
7: $omij^d{+}{+}$ all i to n;
8: $j{+}{+}$;
9: **end if**
10: $i{+}{+}$;
11: **end for**
12: **return** OM^d;

Definition 3. Ordered relation: let A and B be two non-empty subsets in the given binary partition; an ordered relation between A and B is defined as follows: $A \leq B$, if for all a in A, all b in B such that $a.u < b.l$.

Through the binary partition, if two regions are in different partitions, they do not overlap through the ordered relation of the partitions. As can be seen in the definition above, the ordered relation between different partitions plays an important role in deciding the overall overlap information of regions. The algorithm constructs partitions of regions. As there is no need to determine the overlap information between the left-hand and right-hand partitions' regions, the algorithm performs the matching operation for the pivot set with the other set in the left-hand and right-hand partitions. In other words, it is not necessary to compare and match different partitions through the ordered relation in the binary partition.

From the binary partition, we can extract regions that fall on the boundaries of partitions by using BinaryPartition(). These regions are collected into the pivot set, which consists of two subsets of regions, S_{pu} and S_{ps}. These sets are used for the matching process to compare

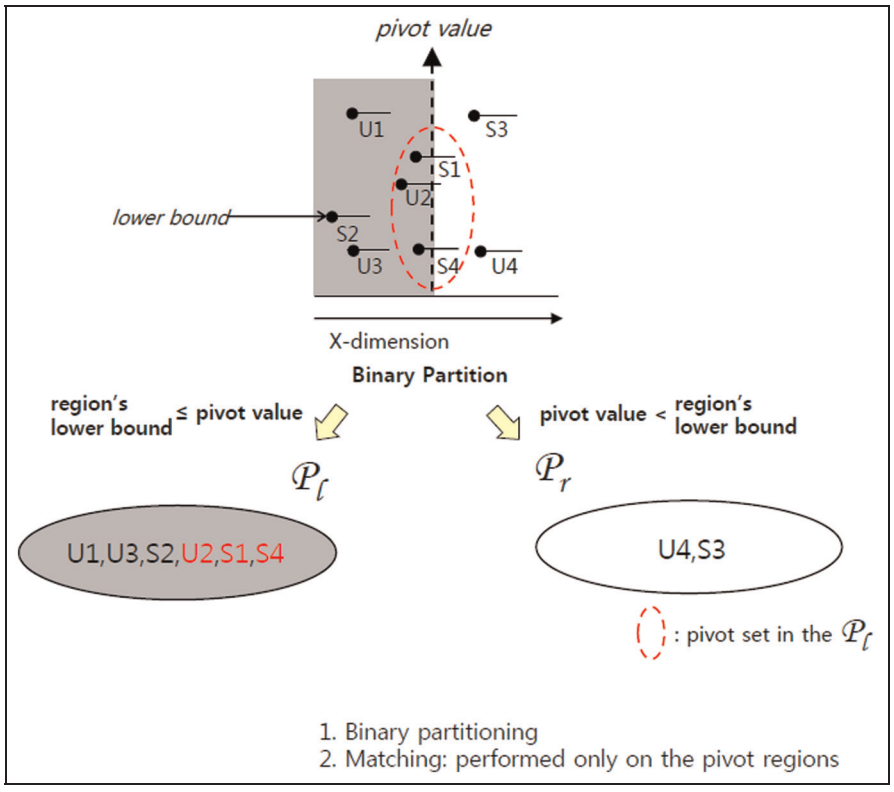

Figure 3. Binary partition into the left-hand and right-hand partitions, P_l and P_r.

the intersection between regions in the right-hand partition, $\mathcal{P}_r$.

$$(U(i) \cdot l < S(j).u) \;\&\; (S(j).l < U(i).u) \tag{1}$$

where *i, j* are region identifiers.

The matching process is checked by the Intersection Calculation() algorithm, which is presented in Algorithm 2. Consider one update region, denoted as U in Figure 4, and one subscription region, denoted as S in the Figure 4. The matching process checks the overlapping of update regions in S_l and subscription regions in S_u. As described in the HLA Interface Specification, a range is specified as a continuous semi open interval (lower bound, upper bound) on a dimension. After a dimension is projected in the *x*-dimension, the necessary and sufficient condition for the two ranges to overlap is satisfaction of inequality (1). If the two ranges overlap, it is true that two regions are satisfied by inequality (1), $(U(i).l < S(j).u)$ and $(S(j).l < U(i).u)$.

The IntersectionCalculation() algorithm determines whether subscription regions in S_{ps} overlap with update regions in S_{lu} and S_{ru} or update regions in S_{pu} overlap with subscription regions in S_{ls} and S_{rs} because of the ordered relation. In addition, all update regions and subscription regions in the pivot set are overlapped, since their range includes the pivot value. As the regions in the boundary of the pivot set will all be overlapped, the proposed algorithm easily achieves the calculation of the intersection in the matching process. If there is a distribution with large overlapping regions, the calculation of the intersection is easily obtained. Therefore, this approach is more efficient in computing the large overlapping regions in each partition. This matching process results in a matrix with overlap information, OM. The overlap matrix is the *n n* matrix (*n* * *n* the number of regions). If two regions whose handles are (*i, j*) are overlapped in the *d* dimension, $omij^d$ is added (+1). After the whole matching process, if *omij* in OM is the same as the number of dimensions, *d*, two regions $\mathcal{R}_i$ and $\mathcal{R}_j$ are exactly overlapped. In the example explained in Table 3, we obtain $OMx = \{ou1; s2; ou2; s1; ou2; s4; ou3; s2; ou4; s3\}$ on the *x*-dimension.

As described in Section 4.1, our proposed algorithm decomposes the multidimensional space into a number of partitions through binary partitioning. The algorithm adopts the divide-and-conquer strategy based on the ordered relation. The flowchart of the proposed algorithm is presented in Figure 5. The process of the flowchart is executed in the following steps.

1) Project all dimensions of regions in the multidimensional space into one dimension through the dimension projection algorithm.

2) Set a midpoint as the pivot value in the projected regions using Algorithm 1.
3) Divide the projected regions into two partitions, $\mathcal{P}_l$ and $\mathcal{P}_r$, with six sets, S_{lu}, S_{ls}, S_{ru}, S_{rs}, S_{pu}, and S_{ps}, through the binary partition.
4) Perform the matching process in the pivot set. Also, perform the matching process between different partitions to compare the update regions in S_{pu} with the subscription regions in S_{rs} and S_{ls} and the subscription regions in S_{ps} with the update regions in S_{ru} and S_{us} using Algorithm 2. From this matching process, our algorithm determines OM, which stores the overlap information into an n-by-n matrix (where n is the number of regions).
5) Iterate the pivot selection and matching process in $\mathcal{P}_l$ and $\mathcal{P}_r$ until the remainder is never partitioned.

The binary partition-based matching algorithm is shown in Algorithm 3. It uses two procedures, BinaryPartition() using Algorithm 1 and IntersectionCalculation() using Algorithm 2, for the partition process and the matching process. BinaryPartition() is a recursive procedure that divides regions into two partitions, and IntersectionCalculation() is a procedure that checks the intersection between regions in the partitions, through the ordered relation. We assume that there are regions, $\mathbb{R}$, and $g1$ is the minimum lower bound and $g2$ the maximum upper bound in $\mathbb{R}$. Next, the regions are projected into the $\mathsf{T}\mathbb{R}^d$ that represent the regions of the corresponding dimension, d.

During the binary partition process, the left-hand and right-hand partitions maintain six sets, S_{lu}, S_{ru}, S_{pu}, S_{ls}, S_{rs}, and S_{ps}, through BinaryPartition(). After that, it is checked by IntersectionCalculation() to determine whether subscription regions in S_{ps} overlap with update regions in S_{lu} and S_{ru} or update regions in S_{pu} overlap with the subscription regions in S_{ls} and S_{rs} through the ordered relation. In addition, all update regions and subscription regions in the pivot set overlap, as these regions fall inside the boundaries of the partitions. It repeats these processes for multiple rounds until all overlap information is constructed. Finally, the OM stores the overall overlapping status for all dimensions.

4.4 Theoretical analysis on computational complexity

To analyze the computational complexity of the proposed algorithm, we suppose that there are N regions. For the simple analysis provided here, we assume that the number of dimensions d is two in the multidimensional space. Clearly, the runtime depends on the number of regions N and of used dimensions d.

The process of the binary partitioning of each dimension is based on a particular pivot value. Steps 6–9 in Algorithm 3 require an $\mathcal{O}$ (N) computation on average using the BinaryPartition() procedure. Steps 21 and 22 to find the exact position of a region in the partitions in Algorithm 3 require an $\mathcal{O}$ (log N) computation by the binary partitioning. Steps 5–11 of the IntersectionCalculation() procedure require an $\mathcal{O}(n)$ computation by comparing the intersection of regions between two subset of partitions (where n is the number of regions in a partition). Because the overlap information of all regions is obtained by the pivot set, it is not necessary to compare their overlap information in the left-hand and right-hand partitions because of the ordered relation. In cases where there is no overlapping at the binary partitioning with the left-hand and right-hand partitions, the matching process is performed. After processing one dimension, Steps 5–23 of this binary partition-based matching procedure are repeated for all the other dimensions. The total computational complexity of the proposed algorithm is $d * (N + n) * \mathcal{O}$ (log N), which increases proportionally with the number of dimensions d. Therefore, the computational complexity of the proposed algorithm is a $d * N * \mathcal{O}$(log N) computation. However, the actual computational complexity depends on how well the binary partition is achieved.

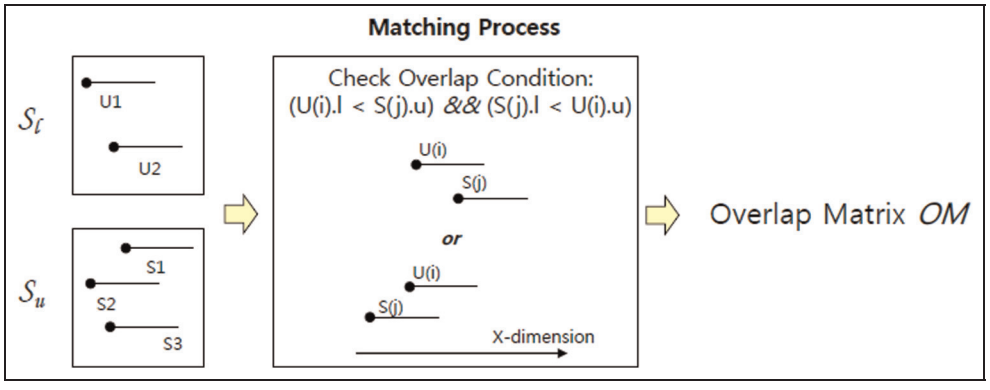

Figure 4. Matching process between S_l and S_r.

Algorithm 3. Binary partition-based matching algorithm.

Input: Regions $\mathbb{R}$; dimension d
Output: n-by-n matrix, OM = (om_{ij}), where $i, j \in n$
1: **procedure** BinaryPartitionbasedMatchingAlgorithm(*Region* $\mathbb{R}$)
2: **if** *Region* $\mathbb{R}$ = Ø **then**
3: **return**;
4: **else**
5: **for** Each dimension, d **do**
6: $g1 \leftarrow$ the minimum lower bound of d dimension;
7: $g2 \leftarrow$ the maximum upper bound of d dimension;
8: Set a pivot value, $p \leftarrow (g1 + g2)/2$;
9: Use the BinaryPartition($T\mathbb{R}^d$, p) to find six sets S_{lu}, S_{ru}, S_{pu}, S_{ls}, S_{rs}, S_{ps};
{*For comparison the pivot set itself*}
10: **for** Each region $Ri \in S_{pu}$ **do**
11: **for** Each region $Rj \in S_{ps}$ **do**
12: $omij^d + +$;
13: **end for**
14: **end for**
{*For comparison with the other set in the left-hand and the right-hand partitions*}
15: OM^d = IntersectionCalculation(S_{pu}, S_{rs});
16: OM^d = IntersectionCalculation(S_{ps}, S_{ru});
17: OM^d = IntersectionCalculation(S_{lu}, S_{ps});
18: OM^d = IntersectionCalculation(S_{ls}, S_{pu});
19: $S_l \leftarrow S_{lu} \bigcup S_{ls}$;
20: $S_r \leftarrow S_{ru} \bigcup S_{rs}$;
21: BinaryPartitionbasedMatchingAlgorithm(Region S_l);
22: BinaryPartitionbasedMatchingAlgorithm(Region S_r);
23: $OM = OM + OM^d$;
24: **end for**
25: **end if**

5. Experimental evaluation

This section presents an experimental evaluation of the proposed method. We compare the proposed algorithm with the previous DDM-matching algorithms in the literature, the region-based algorithm, hybrid approach,[31] and Raczy's sort-based algorithm.[7] We implemented these algorithms with the C++ language and processed them in HLA-based distributed simulations. Our experiments were conducted using Microsoft Windows 7 with a 2.80 GHz Intel(R) Core(TM) i7 central processing unit (CPU) and 8 GB of memory. We assume that our experimental environment has a two-dimensional space for the sake of clarity, as described in Section 4. Since the performance of the hybrid approach depends on the number of grid cells, we used configurations of both 10 * 10 and 100 * 100 grid cells, as Raczy et al.[7] and Ayani et al.[31]

There are several parameters that vary and affect the performance of a matching algorithm. One of the important experimental parameters is the number of regions. This parameter is capable of characterizing the entire situation of the regions for the matching process. It represents the scalability of the system. Next, we also use two different methods to generate the location of each region in the space. These methods have a uniform distribution and clustered distribution, as described in Figure 6(a) and (b). As these synthetic regions can be useful in a variety of situations, our experiment provides realistic region distribution to third parties for testing the DDM-matching algorithms. Therefore, the experiment uses a uniform distribution and clustered distribution as the synthetic region distribution, which can achieve simple forms of the realistic data set. In the uniform distribution, the regions are distributed randomly across the space, whereas the clustered distribution has regions of around $k \geq 1$ clusters.

Finally, the overlap rate is defined as the proportion of the scene volume occupied by the regions. There can be many regions that nearly cover the space without overlapping or there can be very few regions that are tightly overlapping that cover very little of the space. However, it is not possible to accurately measure the number of overlapping regions. We use the potential region overlaps, not the actual region overlaps. The formulation of the overlap rate is the percentage of the occupied regions in the total area of space, which mean the number of potential region overlaps, as follows:

$$Overlap\ rate \frac{\sum area\ of\ region}{area\ of\ space} \tag{2}$$

If the space is 100*100 and one region is 1*1, where the number of regions is fixed at 100, the overlap rate is 0.01 = $\frac{100*(1*1)}{100*100}$. Note that a higher overlap rate potentially implies a greater probability of region overlap. We use the experimental parameters and related values summarized in Table 4. For enhanced accuracy and reliability, we repeat the experiment more than 30 times. A 95% confidence interval was estimated for all experimental results.

To evaluate the performance of the proposed algorithm in terms of the dynamic range measure, as well as the execution time for the matching process, two sets of results are presented. Firstly, we compare the execution time (*ET*) in executing the runtime required for calculating the intersections for the matching process. Secondly, experiments are carried out to test a normalized dynamic range (*NDR*), which is the ratio of the dynamic range value to the overlap rate. In order to reflect the region modification over a range of overlapping conditions, we define the *NDR* as the

Table 4. Three partitions of the binary partition.

Parameters	Value range
The usage of the matching algorithm	Our algorithm, region, sort, hybrid
# of regions	1000–10,000
Region distribution	Uniform and clustered
Overlap rate	0.01, 0.1, and 1

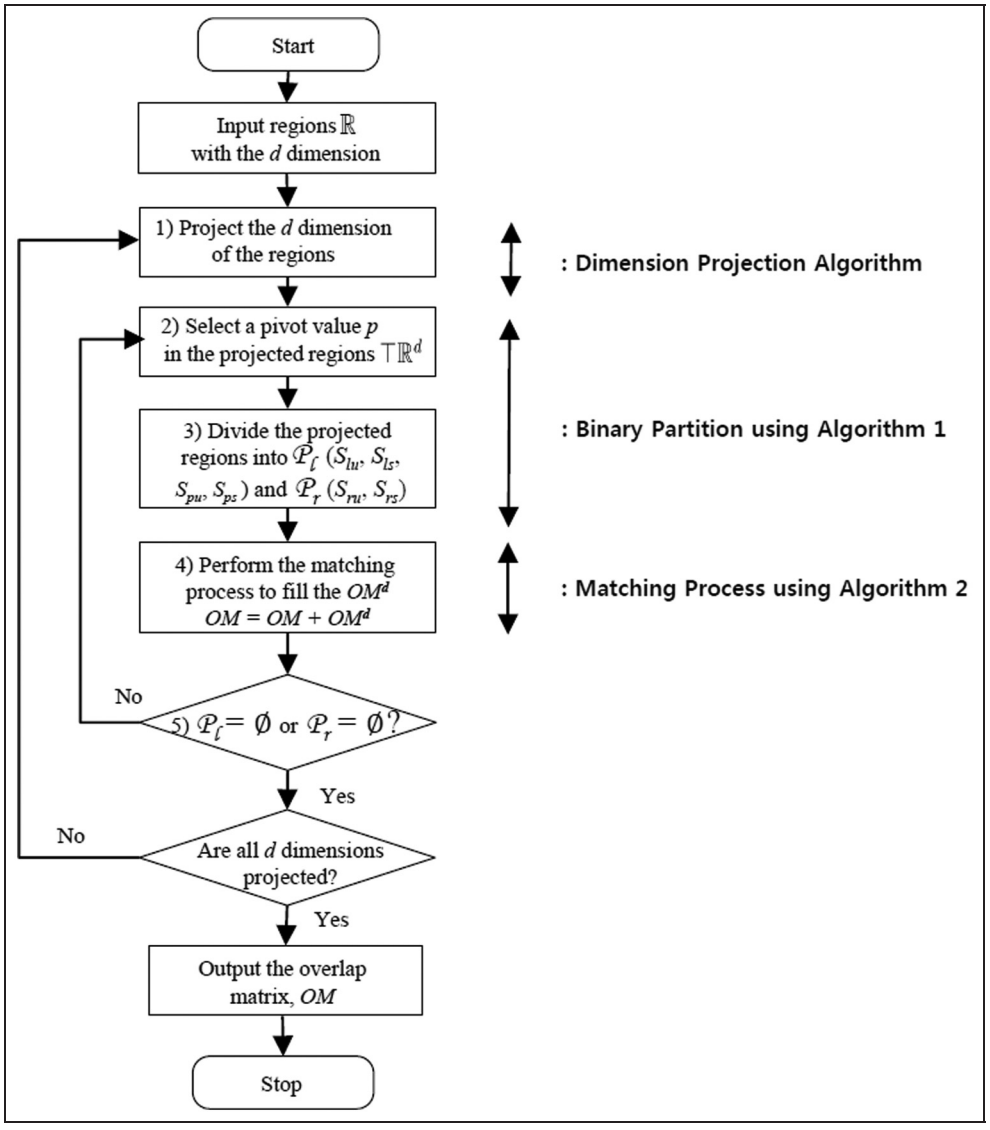

Figure 5. Flowchart of the binary partition-based matching process using Algorithm 3.

ratio of the variance of the execution time to the current execution time at the specific overlap rate, which causes this variance. It means the robustness of the performance of algorithms against changing of the overlapping regions, which is able to deal with region modifications during execution. *NDR* is particularly useful to understand how a matching algorithm depends on variations in the parameter of the overlap rate. The *NDR* can be expressed as

$$NDR = \frac{ETcurrent\ overlap\ rate - ETprevious\ overlap\ rate}{ETcurrentoverlap\ rate} * 100(\%) \quad (3)$$

Using Equation (3), the NDR is calculated when the number of region = 5000 is fixed. We observe that the region-based algorithm has same execution time although the overlap rate varies with same. We observe that the region-based algorithm always has same execution time when the overlap rate varies at the same number of regionsumber of regions. The region-based algorithm is the most robust. Thus, *NDR* = 0 is not presented in the following comparisons. In our experiments, this equation is useful to test how robust the algorithm is to changes in the overlap rate.

5.1 Region distribution

5.1.1 Uniform distribution. Figure 7 shows the execution time for the matching process in four algorithms when the overlap rate is 0.01. It seems that the hybrid approach with 100 * 100 grid cells always have the best performance, because the regions are small in one cell. However, the hybrid algorithm did not perform well in situations where

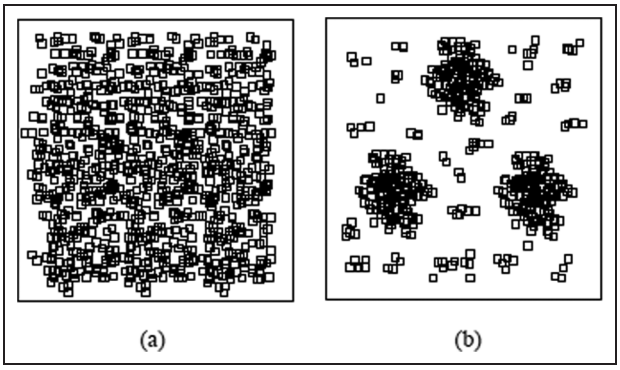

Figure 6. Space wherein regions are scattered under (a) uniform and (b) clustered distribution.

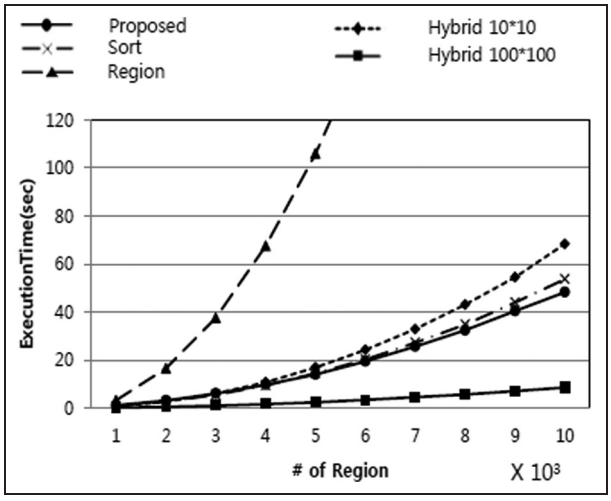

Figure 7. Performance comparisons: overlap rate = 0.01 under uniform distribution.

the size of the grid cells changed (i.e. the performance for the 10 * 10 grid cells of the hybrid approach is degraded). The proposed algorithm outperforms the other algorithms when the overlap rate is 0.01 under the uniform distribution (except the hybrid approach, where the overlap rate is low and the optimal 10 * 10 grid cells are used).

Figure 8 shows the execution time for the matching process when the overlap rate is 0.1. According to this figure, we can see that the proposed algorithm usually performs well in this situation. The performance of the proposed algorithm is affected by the partitioning of the regions in the different partitions. The other algorithms have the same result when the overlap rate is 0.01.

When the overlap rate = 1, according to Figure 9, the proposed algorithm performs well. The proposed algorithm is not the best choice when the overlap is relatively low, but it has the advantage when the overlap rate is high. The more regions overlap with the pivot set, the less matching overhead is incurred in the whole matching

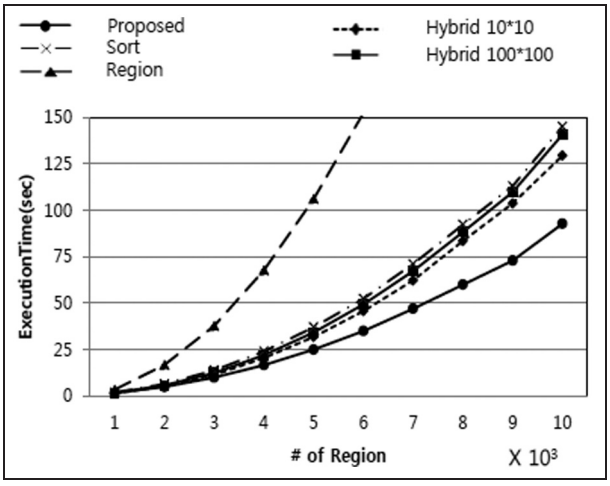

Figure 8. Performance comparisons: overlap rate = 0.1 under uniform distribution.

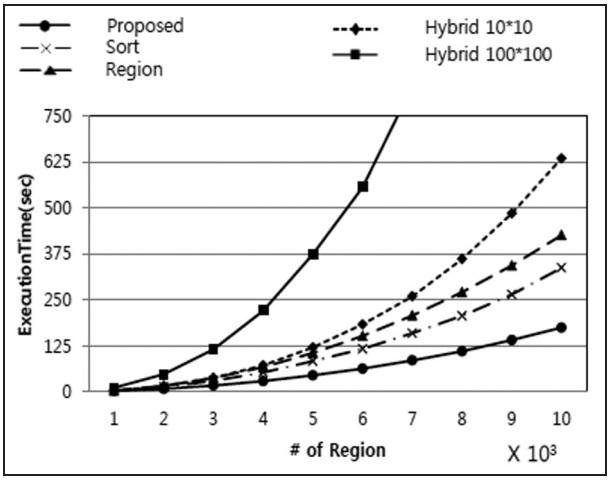

Figure 9. Performance comparisons: overlap rate = 1 under uniform distribution.

process. This means that the more regions are overlapped, the better the performance of the proposed algorithm will be. On the other hand, the performance of the sort-based algorithm is much better than the region-based algorithm and the hybrid approach, except for ours. When the number of regions increases and the overlap rate is high, the performance of the region-based algorithm becomes increasingly better than with the other overlap rate. The hybrid approach has a higher computational overhead, particularly with 100 * 100 grid cells. This approach degrades performance when the overlap rate is high. According to all of the figures, we know that the hybrid algorithm with 100 * 100 grid cells has an extremely large computational overhead for the matching process. In general, the proposed algorithm outperforms, on average, by over 35% in some situations with a high overlap rate and a uniform distribution.

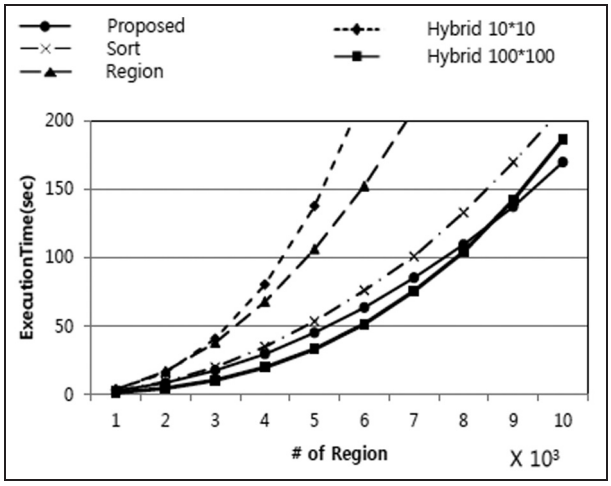

Figure 10. Performance comparisons: overlap rate = 0.01 under clustered distribution.

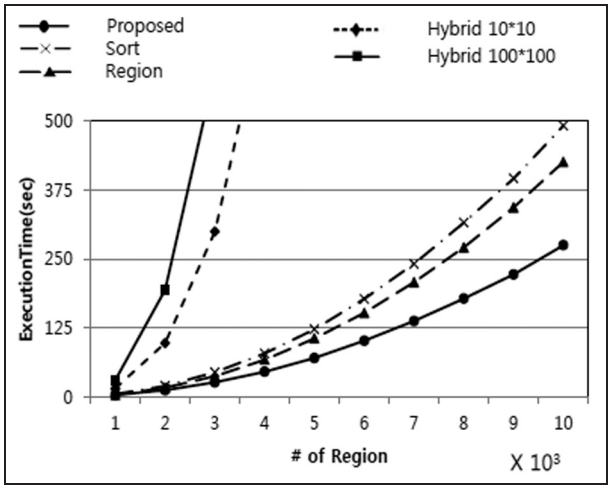

Figure 11. Performance comparisons: overlap rate = 0.1 under clustered distribution.

5.1.2 Clustered distribution. We next varied the overlap rate under the clustered distribution. The second set of experiments compared the execution time for the matching process under the clustered distribution. When the regions are distributed under the clustered distribution, there is a greater probability of region overlap than for the uniform distribution in the fixed number of regions. Figure 10 shows the performance comparison of the four algorithms. The hybrid algorithm with 10 * 10 yields poor performance compared with the other matching algorithms. When the number of regions was more than 9000, the proposed algorithm performed better than the hybrid algorithm with 100 * 100. This finding suggests that the proposed algorithm is scalable with increasing region size and can be used for a large-scale spatial environment.

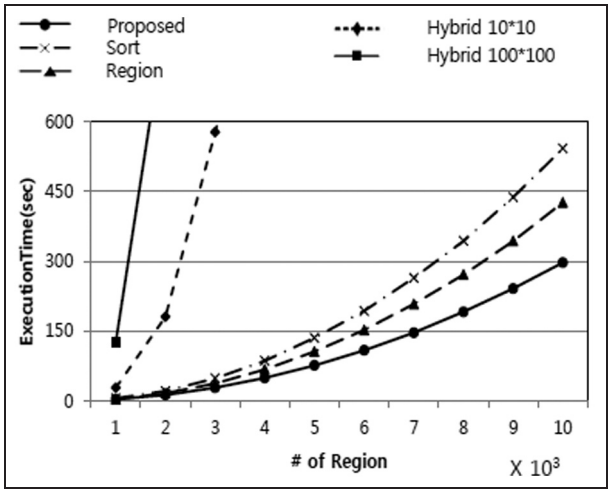

Figure 12. Performance comparisons: overlap rate = 0.1 under clustered distribution.

As described in Figure 11, the hybrid approach is not much more scalable. For this same reason, the hybrid approach checks a pair of update regions and subscription regions within each cell. If regions overlap at a higher rate, the performance of the hybrid approach is much worse than the others. Even worse, they will not perform well in the clustered distribution, since most regions are collected at particular points. The region-based algorithm and the sort-based algorithm perform well when there is a high probability of finding a pair of regions that intersect. Our proposed algorithm is the best choice under the clustered distribution. Since the regions in the boundary of the same partition will all be overlapped, the proposed algorithm can easily find the calculation of the intersection for the matching process from the binary partitioning. This results in performance improvement in some partitions, especially with a high rate of overlapping regions.

Figure 12 shows the execution time for the matching process. All experimental results are similar when the overlap rate is 0.1. The proposed algorithm shows the best performance compared with the other algorithms, as expected. This shows that our proposed algorithm is scalable and performs well under the clustered distribution. On average, the proposed algorithm significantly outperforms the previous algorithms by more than 45% when many regions overlap in the clustered distribution.

5.2 Normalized dynamic range

As can be seen in Table 5, all *NDR*s under the uniform distribution decrease with an increase in the current overlap rate. This means that a higher overlap rate is associated with a lower *NDR* after adjusting for the overlap rate. The *NDR* ranges from 46 to 50% for the hybrid approach with

Table 5. Normalized dynamic range at 5000 regions under uniform distribution.

Overlap rate interval	Hybrid 10 _ 10	Hybrid 100 _ 100	Sort	Proposed
0.01–0.1	46	92	60	43
0.1–0.5	50	78	36	27
0.5–1	46	56	30	24

Table 6. Normalized dynamic range at 5000 regions under clustered distribution.

Overlap rate interval	Hybrid 10 _ 10	Hybrid 100 _ 100	Sort	Proposed
0.01–0.1	85	98	56	36
0.1–0.5	40	43	7	4
0.5–1	35	37	5	3

10 * 10, 56 to 92% for the hybrid approach with 100 * 100, 30 to 60% for the sort-based algorithm, and 24 to 43% for the proposed algorithm. Thus, the proposed algorithm is more robust to variations in the overlap rate than the others under the uniform distribution in terms of the *NDR*.

Next, the clustered distribution is applied to compare the *NDR* with the same parameters as in the previous experiment for the dynamic range measure. As the clustered distribution has a higher overlap rate in some points and this implies a greater probability of region overlap, Table 6 has the same trend as a result of uniform distribution's *NDR*, but it is slightly different. We observe that the *NDR* under the clustered distribution strictly decreases when the overlap rate increases. In addition, the variance of the hybrid approach is quite large, and the *NDR* is higher than the others.

When the overlap rate increases from 0.1 to 1, the variation of the *NDR* changes, especially in the sort-based algorithm and the proposed algorithm. However, the variance of the proposed algorithm (i.e. 3–36%) is smaller than that of the sort-based algorithm (i.e. 5–56%). Therefore, the proposed algorithm has robustness over the range of overlapping conditions under the clustered distribution.

5.3 Discussion

(1) The region-based algorithm is a brute-force method that is simple and directly implemented. As mentioned above, the region-based algorithm checks all possible combinations of regions to construct the overlap information. In such contexts, this algorithm is too time consuming and can be impractical. In spite of the extensive amount of computation, when there is a situation of region distribution at a high overlap rate or under the clustered distribution, the region-based algorithm performs well compared to the other matching algorithms, except our proposed algorithm. In addition, this approach is more robust than the other matching algorithms.

(2) The hybrid approach is based on a grid-based algorithm, which is adopted by the brute-force method in each grid cell to compute the intersections. It has the best performance at a low overlap rate when the optimal grid size is used. As the hybrid approach checks a pair of update regions and subscription regions within each cell, the performance of the hybrid approach is much worse than the others at a high overlap rate. The hybrid approach's performance becomes worse when the cells are too small in the uniform distribution of regions. There is no easy solution to compute the ideal grid cell's size, which affects the algorithm's behavior. Even worse, it will not perform very well in situations where the distribution of regions is clustered. Moreover, the hybrid approach has low *NDR* compared to the others.

(3) The sort-based algorithm sorts the regions according to their coordinates in each dimension. On average, its performance is much better than the region-based and hybrid algorithms. It is not the worst choice in some situations in which the characteristics of the region distribution are not identified. However, the sort-based algorithm still has a disadvantage when the overlap rate is high and the distribution of regions is not uniform. Since the sort-based algorithm computes the intersections globally and simultaneously, it can be used to optimize the sorting data structure for an efficient matching operation.

(4) As our algorithm avoids checking any regions between different partitions by using the ordered relation of partitions, it is the best choice (except for the hybrid approach, when the overlap rate is low and the optimal grid size is used). Furthermore, it has additional advantages compared with the other algorithms. Firstly, the proposed algorithm is practical and efficient, because it reduces the computational overhead and improves the overall performance of the matching process. Since the proposed algorithm reflects the region distribution by using the concept of the ordered relation, it easily calculates the intersection between regions on partition boundaries and does not require unnecessary comparisons within

regions in different partitions. Secondly, it is less affected by the overlap rate. The algorithm recursively performs binary partitioning, which divides the regions into two partitions that entirely cover those regions.

This means that there is no need to determine the overlap information between partitions, because they already consider whether the regions are intersected through the ordered relation. This requires minimal computational overhead in some partitions, especially when the rate of overlapping regions is high. This is attributed to the same reason as in the case of the clustered distribution. In addition, the proposed algorithm works more robustly than the other matching algorithms in the *NDR* with an increasing overlap rate (except for the region-based algorithm).

Overall, from the experimental results, we found that our algorithm performs better than the others in a given set of scenarios, especially at a high overlap rate and in a clustered distribution. Furthermore, it improves the scalability of DDM implementation for large-scale simulations. Moreover, to complete the study, we tested all the algorithms in more complex situations, such as scenarios in which the distribution of regions is not uniform (i.e. a clustered distribution). Therefore, the proposed algorithm has the potential to enhance the robustness at high overlap rates.

6. Conclusion

The focus of this paper is on matching algorithms that have been developed for DDM in HLA-based distributed simulations. In recent years, many DDM-matching algorithms have been proposed and investigated to reduce the higher computational overhead from the matching process. However, as the previous works we have examined so far do not identify the characteristics of the region distribution, it is difficult to select a matching algorithm that is appropriate for some region-distribution situations.

Therefore, in this paper, we propose a binary partition-based matching algorithm based on the divide-and-conquer approach. The algorithm recursively performs binary partitioning, which divides the regions into two partitions that entirely cover those regions. The proposed algorithm is, to the best of our knowledge, the first attempt to facilitate binary partitioning and obtain the overlap information using the concept of an ordered relation previously not considered. The algorithm reduces the computational overhead for the matching process, since it easily calculates the intersection between regions on partition boundaries and does not require unnecessary comparisons within regions in different partitions. According to the theoretical analysis, the proposed algorithm has an advantage in scalability. We have developed a binary partition-based matching algorithm in the DDM and compared its performance with the previous matching algorithms in different region-distribution situations. Our experimental results show that the proposed algorithm performs better than the previous matching algorithms across a variety of workloads in terms of the dynamic range measure, as well as the execution time for the matching process. In particular, the proposed algorithm significantly outperforms in a given set of scenarios with a high overlap rate. In more complex scenarios, where the distribution of regions is clustered, not uniform, the proposed algorithm is much better than the others. In addition, it is more robust than the others in terms of the NDR. Moreover, the proposed algorithm improves the scalability of DDM implementation for large-scale spatial simulations. In future works, we plan to optimize an efficient algorithm for the initialization cost of the binary partitioning and improve further the scalability of the proposed algorithm in more complex dynamic scenarios. In addition, the proposed algorithm will be extended for more general applications, such as a real HLA-based distributed application that is more realistic.

Acknowledgment

The authors would like to thank the anonymous referees for their helpful and insightful suggestions.

Funding

This work was partially supported by the Defense Acquisition Program Administration and Agency for Defense Development (contract UD110006MD) and the Brain Korea 21 Project, BK Electronics and Communications Technology Division, KAIST in 2011.

References

1. Morse K and Steinman JS. Data distribution management in HLA: multidimensional regions and physically correct filtering. In: *proceedings of the spring simulation interoperability workshop*, 1997, p.97S-SIW-053.
2. Hook DJV and Calvin JO. Data distribution management in RTI 1.3. In: *proceedings of the fall simulation interoperability workshop*, 1998, p.98F-SIW-206.
3. Morse KL and Petty MD. High level architecture data distribution management migration from DoD 1.3 to IEEE 1516. *Concurrency Pract Ex* 2004; 16: 1527–1543.
4. Petty MD and Morse KL. The computational complexity of the high level architecture data distribution management matching and connecting processes. *Simul Model Pract Theory* 2004; 12: 217–237.
5. Tan GSH, Xu L, Moradi F, et al. An agent-based DDM filtering mechanism. In: *MASCOTS*, 2000, pp.374–381.
6. Boukerche A, Roy A and Thomas N. Dynamic grid-based multicast group assignment in data distribution management.

In: *4th international workshop on distributed simulation and real-time applications (DS-RT 2000)*, 2000, pp.47–54.
7. Raczy C, Tan GSH and Yu J. A sort-based DDM matching algorithm for HLA. *ACM Trans Model Comput Simul* 2005; 15: 14–38.
8. Santoro A and Fujimoto RM. Offloading data distribution management to network processors in HLA-based distributed simulations. *IEEE Trans Parallel Distrib Syst* 2008; 19: 289–298.
9. Dahmann JS and Morse KL. High level architecture for simulation: an update. In: *DIS-RT*, 1998, pp.32–40.
10. IEEE Std 1516-2000. IEEE standard for modeling and simulation (M&S) High Level Architecture (HLA) - framework and rules, 2000.
11. Fujimoto RM. *Parallel and distributed simulation systems*. New York: Wiley Interscience, 2000.
12. Dahmann JS, Fujimoto RM and Weatherly RM. The DoD high level architecture: an update. In: *Winter Simulation Conference*, 1998, pp.797–804.
13. Dahmann JS, Kuhl FS and Weatherly RM. Standards for simulation: as simple as possible but not simpler the high level architecture for simulation. *Simulation* 1998; 71: 378–387.
14. IEEE Std 1516-2010. IEEE standard for modeling and simulation (M&S) High Level Architecture (HLA) - framework and rules, 2010.
15. IEEE Std 1516.1-2010. IEEE standard for modeling and simulation (M&S) High Level Architecture (HLA) - federate interface specification, 2010.
16. IEEE Std 1516.2-2010. IEEE standard for modeling and simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) specification, 2010.
17. DMSO. RTI 1.3-next generation programmer's guide version 5, DoD, February 2002.
18. Lee JS, Zeigler BP and Venkatesan SM. Design and development of data distribution management environment. *Simulation* 2001; 77: 39–52.
19. Lee JS and Zeigler BP. Space-based communication data management in scalable distributed simulation. *J Parallel Distrib Comput* 2002; 62: 336–365.
20. Hild DR, Sarjoughian HS and Zeigler BP. DEVS-DOC: a modeling and simulation environment enabling distributed codesign. *IEEE Trans Syst Man Cybern Part A Syst Humans* 2002; 32: 78–92.
21. Lee JS and Zeigler BP. Dynamic multiplexing and high-performance modeling in distributed simulation. *Simulation* 2005; 81: 365–380.
22. Duong TNB, Zhou S and Shen H. Greedy algorithms for client assignment in large-scale distributed virtual environments. *Simulation* 2008; 84: 521–533.
23. Grande RED and Boukerche A. Dynamic balancing of communication and computation load for HLA-based simulations on large-scale distributed systems. *J Parallel Distrib Comput* 2011; 71: 40–52.
24. Lu T, Lee C, Hsia W, et al. Supporting large-scale distributed simulation using HLA. *ACM Trans Model Comput Simul* 2000; 10: 268–294.
25. Su-Youn Hong J-HK and Kim TG. Measurement of RTI performance for tuning parameters to improve federation performance in real-time war game simulation. In: *proceedings of the 37th summer computer simulation conference (SCSC 2005)*, 2005, pp.71–76.
26. Zhou S, Cai W, Turner SJ, et al. Flexible state update mechanism for large-scale distributed wargame simulations. *Simulation* 2007; 83: 707–719.
27. Tan GSH, Zhang Y and Ayani R. A hybrid approach to data distribution management. In: *4th international workshop on distributed simulation and real-time applications (DS-RT 2000)*, 2000, pp.55–61.
28. Boukerche A and Dzermajko C. Performance evaluation of data distribution management strategies. *Concurrency Pract Ex* 2004; 16: 1545–1573.
29. Boukerche A, Gu Y and de Araujo RB. Performance analysis of an adaptive dynamic grid-based approach to data distribution management. In: *10th IEEE international symposium on distributed simulation and real-time applications (DS-RT 2006)*, 2006, pp.175–184.
30. Boukerche A and Guo Y. An efficient adaptive transmission control scheme for large-scale distributed simulation systems. *IEEE Trans Parallel Distrib Syst* 2009; 20: 246–260.
31. Ayani R, Moradi F and Tan GSH. Optimizing cell-size in grid-based DDM. In: *14st international workshop on principles of advanced and distributed simulation (PADS 2000)*, 2000, pp.93–100.
32. Boukerche A, McGraw NJ, Dzermajko C, et al. Grid-filtered region-based data distribution management in large-scale distributed simulation systems. In: *Annual Simulation Symposium*, 2005, pp.259–266.
33. Pan K, Turner SJ, 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 2007)*, 2007, pp.70–82.
34. Liu ES and Theodoropoulos GK. An approach for parallel interest matching in distributed virtual environments. In: *13th IEEE/ACM international symposium on distributed simulation and real time applications (DS-RT 2009)*, 2009, pp.57–65.
35. Lo S-H, Chung YC and Pai F-P. Offloading region matching of data distribution management with CUDA. In: *2010 International Conference on Intelligent Systems, Modelling and Simulation*, 2010.
36. Wang L, Turner SJ and Wang F. Interest management in agent-based distributed simulations. In: *7th international workshop on distributed simulation and real-time applications (DS-RT 2003)*, 2003, pp.20–29.
37. Eroglu O, Mantar HA and Sevilgen FE. Quadtree-based approach to data distribution management for distributed simulations. In: *SpringSim*, 2008, pp.667–674.
38. Cormen TH, Leiserson CE and Rivest RL. *Introduction to algorithms*. Boston: MIT Press, 1990.
39. Aho AV, Ullman JD and Hopcroft JE. *Data structures and algorithms*. Addison-Wesley, 1983.
40. Liu ES, Yip MK and Yu G. Lucid platform: applying HLA DDM to multiplayer online game middleware. *Comput Entertain* 2006; 4.

Author biographies

Junghyun Ahn received his BS degree in electrical engineering from Pusan National University in 2005 and MS degree from the school of electrical engineering and computer science of the Korea Advanced Institute of Science and Technology (KAIST) in 2007. He is currently a PhD candidate at the department of electrical engineering of the KAIST. His research interests include methodology for modeling and simulation (M&S) of discrete event systems (DEVS) and DDM in HLA/RTI.

Changho Sung received his PhD from the department of electrical engineering at the KAIST, Daejeon, Korea. He has experience in defense M&S. From 2005 to 2009, he participated in the US Republic of Korea combined exercises, such as Ulchi Focus Lens, as a M&S Engineer. His research interests include DEVS modeling, collaborative M&S, distributed simulation, and hybrid simulation. He has been a postdoctoral researcher in the systems modeling simulation laboratory at the KAIST since 2011.

Tag Gon Kim received his PhD in computer engineering with a specialization in systems modeling and simulation from the University of Arizona, Tucson, AZ, in 1988. He was an assistant professor at the department of electrical and computer engineering, University of Kansas, Lawrence, KS, US from 1989 to 1991. He joined the electrical engineering department at the KAIST, Tajeon, Korea in fall, 1991, and has been a full professor at the electrical engineering and computer science (EECS) department since fall, 1998. He was the president of the Korea Society for Simulation (KSS) and the editor-in-chief for Simulation: Transactions for Society for Computer Modeling and Simulation International (SCS). He is a co-author of the text book, Theory of Modeling and Simulation, Academic Press, 2000. He has published about 200 papers in M&S theory and practice in international journals and conference proceedings. He is very active in research and education in defense modeling and simulation in Korea. He was/is a technical advisor for defense M&S at various Korea government organizations, including the Ministry of Defence, the Defence Agency for Technology and Quality (DTAQ), the Korea Institute for Defence Analysis (KIDA), and the Agency for Defence Development (ADD). He developed a tools set, call DEVSimHLA, for HLA-compliant war game models development, which has been used for the development of three military war game models for the Navy, Air Force, and Marines in Korea. He is a Fellow of the SCS, a Senior Member of the IEEE and Eta Kappa Nu.