The Journal of Systems and Software 81 (2008) 2345–2360

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Data access in distributed simulations of multi-agent systems

Dan Chen [a,b,*,1], Roland Ewald [b,d], Georgios K. Theodoropoulos [b], Robert Minson [b], Ton Oguara [b], Michael Lees [c], Brian Logan [c], Adelinde M. Uhrmacher [d]

[a] Institute of Electrical Engineering, Yanshan University, Qinhuangdao, China
[b] School of Computer Science, University of Birmingham, Birmingham, UK
[c] School of Computer Science and IT, University of Nottingham, Nottingham, UK
[d] Department of Computer Science, University of Rostock, Rostock, Germany

ARTICLE INFO

Article history:
Received 13 October 2007
Received in revised form 30 April 2008
Accepted 30 April 2008
Available online 7 May 2008

Keywords:
Multi-agent systems
Complex systems
Distributed simulation
Data management
Range query

ABSTRACT

Distributed simulation has emerged as an important instrument for studying large-scale complex systems. Such systems inherently consist of a large number of components, which operate in a large shared state space interacting with it in highly dynamic and unpredictable ways. Optimising access to the shared state space is crucial for achieving efficient simulation executions. Data accesses may take two forms: locating data according to a set of attribute value ranges (range query) or locating a particular state variable from the given identifier (ID query and update). This paper proposes two alternative routing approaches, namely the address-based approach, which locates data according to their address information, and the range-based approach, whose operation is based on looking up attribute value range information along the paths to the destinations. The two algorithms are discussed and analysed in the context of PDES-MAS, a framework for the distributed simulation of multi-agent systems, which uses a hierarchical infrastructure to manage the shared state space. The paper introduces a generic meta-simulation framework which is used to perform a quantitative comparative analysis of the proposed algorithms under various circumstances.

1. Introduction

The last decade has witnessed an explosion of interest in complex systems, which involve dynamic and unpredictable interactions between large numbers of components including software, hardware devices (such as sensors), and social entities (people or collective bodies). Examples of such systems range from traditional embedded systems, to systems controlling critical infrastructures, such as defence, energy, health, transport and telecommunications, to biological systems, to business applications with decision-making capabilities, to social systems and services, such as e-government, e-learning, etc. The complexity of such systems renders simulation modelling the only viable method to study their properties and analyse their emergent behaviour. Multi-agent systems (MAS) have emerged as a particularly suitable paradigm for modelling complex systems. When embedded in a real system, an MAS is itself a complex system whose properties and emergent behaviour have also to be analysed via simulation (Epstein and Axtell, 1996; Jennings and Wooldridge, 1998).

The application of agent-based simulation to ever more complex problems has placed it in the highly computation intensive world with computational requirements far exceeding the memory and performance capabilities of conventional sequential computer systems. As a result, parallel and distributed simulation emerges as a particularly promising and viable approach to alleviate the simulation bottleneck in the design and analysis of large, complex, agent-based systems.

Amongst the most influential of these approaches, the Logical Process Paradigm seeks to divide the simulation model into a network of concurrent logical processes (LPs), each of which models some object(s) or process(es) in the simulated system. Each LP maintains and processes a portion of the state space of the system and state changes are modelled as time-stamped events in the simulation (Fujimoto, 2000).

In conventional distributed simulations, the shared state is typically small and the processes interact with each other in a small number of well-defined ways. The topology of the simulation is determined by the topology of the simulated system and its decomposition into processes, and is largely static. However, in the case of agent-based systems, which operate in complex

* Corresponding author. Address: Institute of Electrical Engineering, Yanshan University, Qinhuangdao, China.
E-mail addresses: d.chen@cs.bham.ac.uk, chendan@pmail.ntu.edu.sg (D. Chen).

[1] Dan Chen and Roland Ewald had a Postdoctoral Research Fellowship and an Internship with Birmingham, respectively, while this work was undertaken.

0164-1212/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2008.04.041

environments and interact with it in highly dynamic and unpredictable patterns, it is often difficult to determine an appropriate simulation topology a priori. In such systems there is a very large set of shared state variables which could, in principle, be accessed or updated by the processes in the model. Encapsulating the shared state in a single process (i.e., via some centralised scheme) introduces a bottleneck, while distributing it all across the LPs (decentralised, event driven scheme) will typically result in frequent all-to-all communication and broadcasting.

In Logan and Theodoropoulos (2001) we have proposed an approach to manage the shared data in distributed simulations of multi-agent systems (MAS). The approach is based on the notion of spheres of influence (SoI) and uses a hierarchical simulation infrastructure to dynamically decompose and distribute the shared state. This framework has been realised in the context of PDES-MAS,[2] a system for the distributed simulation of multi-agent systems. Management of shared data in distributed simulations needs to address two problems, namely data distribution and data accessing. In Oguara et al. (2005) we addressed the first problem and described data distribution algorithms for the PDES-MAS framework; in Lees et al. (2003, 2005) we discussed synchronization issues that arise from such distributions.

In this paper, we focus on the second problem of data access. Data accesses target both individual data items (referred to as ID queries) and selected data items overlapping given query windows (referred to as range queries). This is a challenging issue, particularly when both the value and the physical distribution of data items are dynamic. The physical distribution of data items refers to the assignment of data items to LPs, which are running in separate threads and could thus be distributed physically over a set of machines. Value distribution, in contrast, characterises the content of data items.

Typically, two basic criteria to locate data items can be identified, namely the physical location of individual data items and their attribute value. This paper describes two new candidate algorithms for data accessing in the context of the PDES-MAS framework, the address-based and the range-based routing, each relying mainly on one of the two aforementioned respective criteria. Data access approaches considerably influence the efficiency of simulation execution and contribute to the complexity of system design. This paper first gives a qualitative comparison of the proposed algorithms and then provides a quantitative analysis of the dynamics of the two solutions. The performance of the candidate algorithms has been analysed with respect to the behaviour of the simulated system as well as the characteristics of the simulation infrastructure. Emphasis has been given to the problem of dynamic range queries. The experimental framework used can provide a substantial reference for system designers to address similar problems.

The two algorithms were first outlined in Ewald et al. (2006) where an initial evaluation was also presented. This paper presents a more detailed description of the algorithms and an extended set of results with an in-depth analysis. The rest of the paper is organized as follows: the PDES-MAS framework is introduced in Section 2. Section 3 discusses the alternative approaches to data accessing. Section 4 provides a qualitative comparison of the proposed routing approaches. Section 5 describes the meta-simulation model which has been developed to study the dynamics of the routing approaches. Section 6 presents the benchmark experiments and results. Related work is briefed in Section 7. Section 8 concludes the paper with a summary and ideas for future work.

2. The PDES-MAS framework

PDES-MAS adopts a standard discrete event simulation approach with optimistic synchronization (Logan and Theodoropoulos, 2001). When constructing multi-agent systems using the framework, an MAS is modelled as a network of LPs (Fig. 1). In particular, each agent is modelled as an *Agent Logical Process* (ALP). An ALP has both private state and shared state. The private state is maintained within the ALP, while the shared state can be accessed (read or updated) by other ALPs in the model. Changes in the state of an ALP that may have a causal impact on other ALPs are referred to as external events and are represented by a time-stamped operation on the shared state.

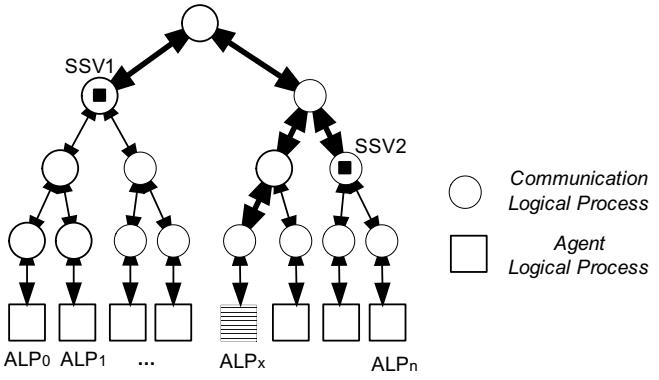

Fig. 1. Overview of the PDES-MAS framework.

The shared state is modelled as a set of *shared state variables* (*SSVs*), each of which is a tuple of the form ⟨*SSV ID*, *attribute type*, *value*, *timestamp*⟩. In PDES-MAS, the shared state is maintained by a tree-structured set of additional logical processes, *communication logical processes* (CLP), which cluster agent models and shared state according to the agents' SoIs (Logan and Theodoropoulos, 2001). As the access patterns on the shared state change, so does the configuration of the tree and the distribution of state (i.e., its allocation to CLPs) to reflect the logical topology of the model.

Redistribution of shared state can be achieved in a number of ways, such as by creating/deleting CLPs, by migrating ALPs through the tree, or by migrating state between CLPs. For the purposes of this paper, we choose to use a fixed tree of CLPs and move SSVs through the tree to achieve redistribution. SSVs are constantly moved closer to the ALPs that access them most frequently, reducing the total access cost and thus contributing to the scalability of the framework (Oguara et al., 2005). The framework does not make use of proxies and only one instance of an SSV is present in the tree at any particular moment. ALPs always link to the leaf CLP nodes in the tree.

The CLP tree provides common services to the ALPs, which includes: (a) facilitating the construction of the distributed simulation; (b) clustering and interoperating the ALPs; (c) managing shared data and balancing load incurred by accessing the shared state; and (d) facilitating synchronization of the ALPs.

Fig. 2 illustrates the relationship between an ALP and the CLP tree. The operation of the CLP tree remains transparent to the ALPs during the simulation. The PDES-MAS framework provides a software library to the ALPs to interact with the CLP tree through two interface modules, referred to as *SimulationAmbassador* and *AgentAmbassador*. An ALP issues requests to access shared state variables through the *SimulationAmassador* module which forwards the requests to the parent (or the server) CLP. If the required SSV is not held locally, the server CLP passes the request to its parent CLP to deal with the request. The return data and control messages (i.e., rollback) are conveyed to the ALP via its *AgentAmbassador* module.

Fig. 3 gives a schematic view of a CLP, which interacts with other LPs in the system via ports. Ports link the individual LPs together to form the overall PDES-MAS simulation system. In this

[2] http://www.cs.bham.ac.uk/research/pdesmas.

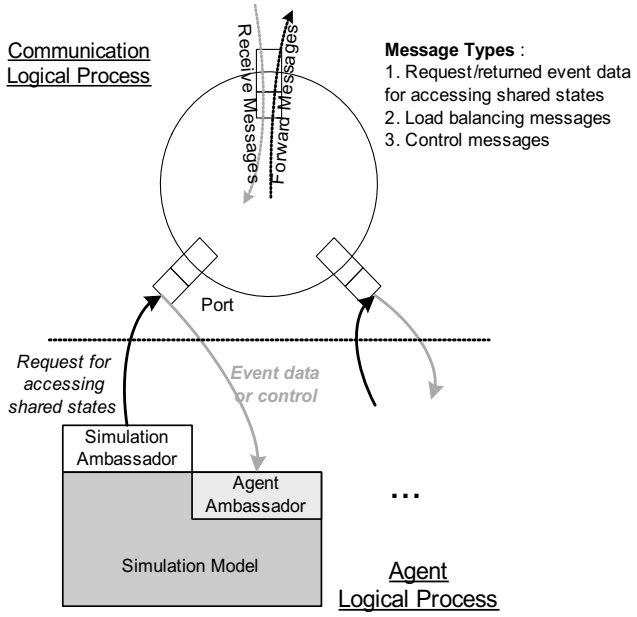

Fig. 2. Relationship between the CLP tree (hierarchical simulation infrastructure) and ALPs.

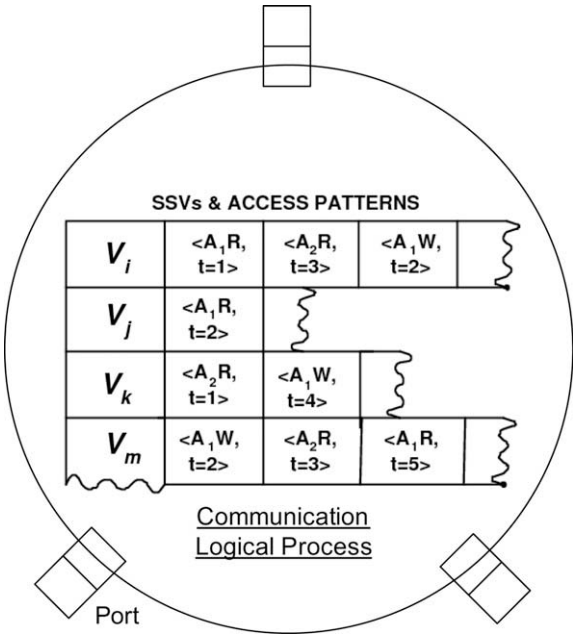

Fig. 3. Communication logical process and ports.

paper, the *incoming* port of a CLP is referred to as the one from which a request on accessing SSV is received. Each CLP is also a router responsible for forwarding access requests to the destination CLP(s) that host the target data, and forwarding is via the *outgoing* ports.

The port is specially designed to maintain the distribution of the values of SSVs in the value space[3] classified by the types of SSVs. The *attribute value range* denotes a certain range of the values of a set of SSVs associated with a particular attribute (or a set of attributes). The "extent" covered by the attribute value range may also vary with different routing algorithms. The distribution concerns SSVs in the local CLP and/or SSVs in remote CLPs, for instance the overall system beyond a port or only the direct neighbours (parent and/or children nodes if any) to the CLP through a port (i.e., as in Figs. 5 and 8).

The distribution of the values gives a panorama of the status of SSVs in the system. No matter what resolution and extent are chosen for an attribute value range, it should always correctly reflect the status of SSVs and can be refreshed once the status changes. More details of attribute value range are available in Section 3.

3. Data access in PDES-MAS

Routing approaches are needed for ALPs and CLPs to locate (a) SSVs according to the attribute value ranges (range query) and/or (b) a particular SSV from the given ID (ID query and update). The two types of locating differ from each other significantly. A range query searches for a group of SSVs based on the specified common attributes and constraints, and the targets tend to differ in each query. This is similar to multicast on dynamic groups. In contrast, ID query aims to locate a unique SSV given its ID, and therefore is closer to a unicast operation.

Routing access requests to SSVs can be performed via either their location information in the tree or the attribute value range information maintained at the ports of the CLPs. SSVs may be moved between CLPs, but there are no multiple copies of a single SSV in the overall system. The status of an SSV can be altered by updating and load management (Oguara et al., 2005). An update may change the value of the SSV, which directly affects the corresponding attribute value range. Load management may induce the migration of the SSV to a different location in the CLP tree, thus, changing the value ranges at related ports. This immediately influences ID queries on this SSV and possibly range queries.

To facilitate the location of SSVs in the CLP tree, it is helpful to encode[4] the tree to identify the CLPs. The address of an SSV is defined as the code of the CLP at which it is maintained (the CLP is referred to as the *host* CLP of this SSV, while the SSV is considered *local* to this CLP). When forwarding access requests, a CLP decides through which port to push the access request to the destination CLP.

The fixed architecture of a CLP tree determines that (a) an SSV can only migrate from a CLP to its direct neighbours, and (b) between any ALP and CLP, there exists only one unique path. Once the target SSVs are located, the returned values need to be simply propagated along the path which the query just traversed in the reverse direction to the source ALP. This section presents two candidate approaches to route ID and Range queries through the tree of CLPs. The two approaches dynamically adapt to different properties of the shared state and the system.

To store the information efficiently, the overall value range of each SSV type is divided into a number of *segments*. Hence, only one bit per segment is needed to store information about the existence of SSVs with values covered by this segment. For example in the case of a CLP containing a set of SSVs with values listed as {20, 53, 56, 70, 80, 190, 310, 370} (see Fig. 13), instead of using a simple range description, such as [Min(20), Max(370)], the value space can be segmented as Seg1: [0, 100], Seg2: [100, 200], Seg3: [200, 300], Seg4: [300, 400], Seg5: [400, 500], ... The approach logs the number of SSVs whose values fall onto each segment. For Fig. 4A, the following segments are defined: {Seg1(5), Seg2(1), Seg3(0), Seg4(2), Seg5(0)}; the attribute range is {Seg1 ∪ Seg2 ∪ Seg4}. When an update occurs, even if the value of the updated SSV is beyond the original [Min, Max], the segmented range may not need to change. For instance, if the SSV with value 370 is updated to

[3] For example, we define "*X*-position" in an extent [0, 100]. Given 100 SSVs of *X*-position, the values of these SSVs may distribute evenly from 0 to 99, such as (0, 1, ..., 99), or concentrate on [50, 51].

[4] No particular coding scheme is preferred as long as it identifies each CLP uniquely and remains consistent in the simulation.

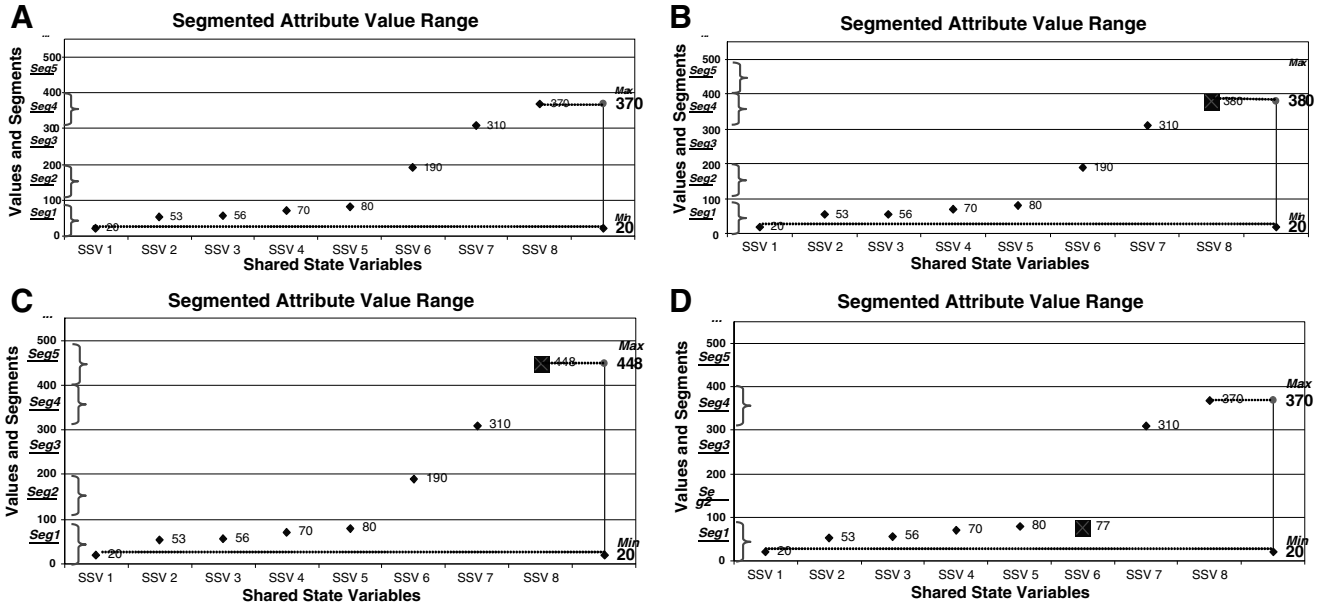

Fig. 4. Segmenting attribute value range.

380, the updated value still falls onto Seg4 (Fig. 4B). The range has to be updated only when there is no SSV having a value in one of the segments (narrowing, Fig. 4D) or any SSV's value exceeds all existing segments (expanding Fig. 4C).

3.1. Address-based routing

The address-based routing searches for SSVs according to their addresses, namely their exact location (host CLP) in the tree. Fig. 5 illustrates an address-based routing approach, which binds the ID of an SSV to its address. Each server CLP maintains a routing table containing the addresses of SSVs that have been accessed in the past. The routing table has a hierarchical format using attribute IDs as indices. From a particular object attribute's perspective, the table maintains the addresses of CLPs hosting the same type of SSV. The SSV IDs of this type are recorded under the host CLP entry. Furthermore, each CLP stores information about the values of SSVs that are hosted by its immediate neighbours. This information is obtained and refreshed when updates on those SSVs occur.

3.1.1. Range query with address-based routing

When an ALP issues a Range query (see example in Fig. 5), its server CLP propagates the request to all CLPs which host SSVs of that SSV type (in this example, CLP_1 and CLP_2). If the values of its SSVs are not within the segments covered by the Range query, the neighbours of a CLP can stop the query (unless there is another CLP that needs to be reached).

The address-based routing algorithm is illustrated in Fig. 6. When a sever CLP receives a range query request for accessing SSVs of a particular type, it needs to resolve the addresses of these SSVs

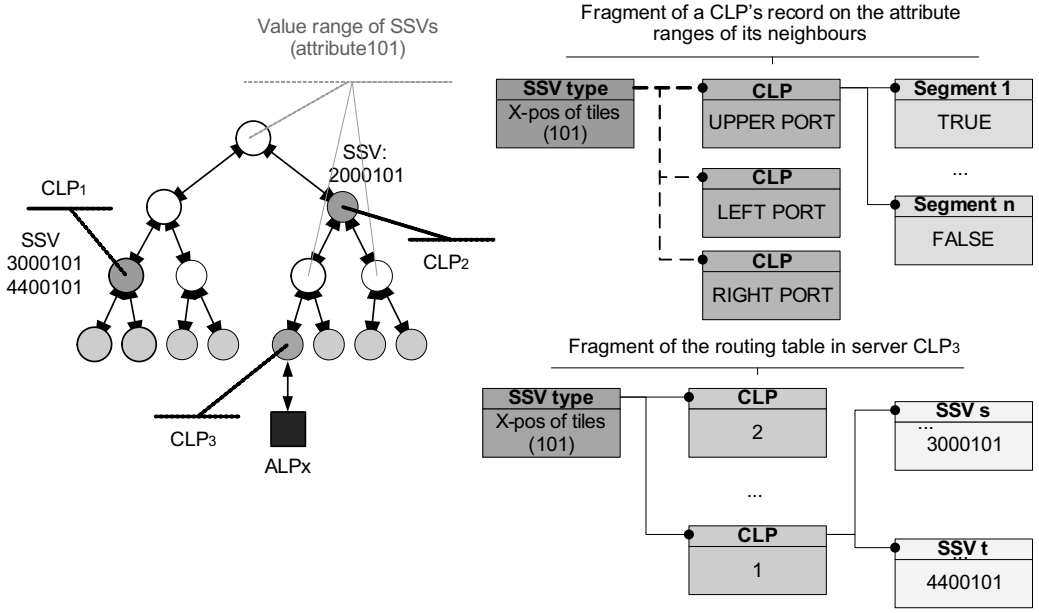

Fig. 5. Address-based routing.

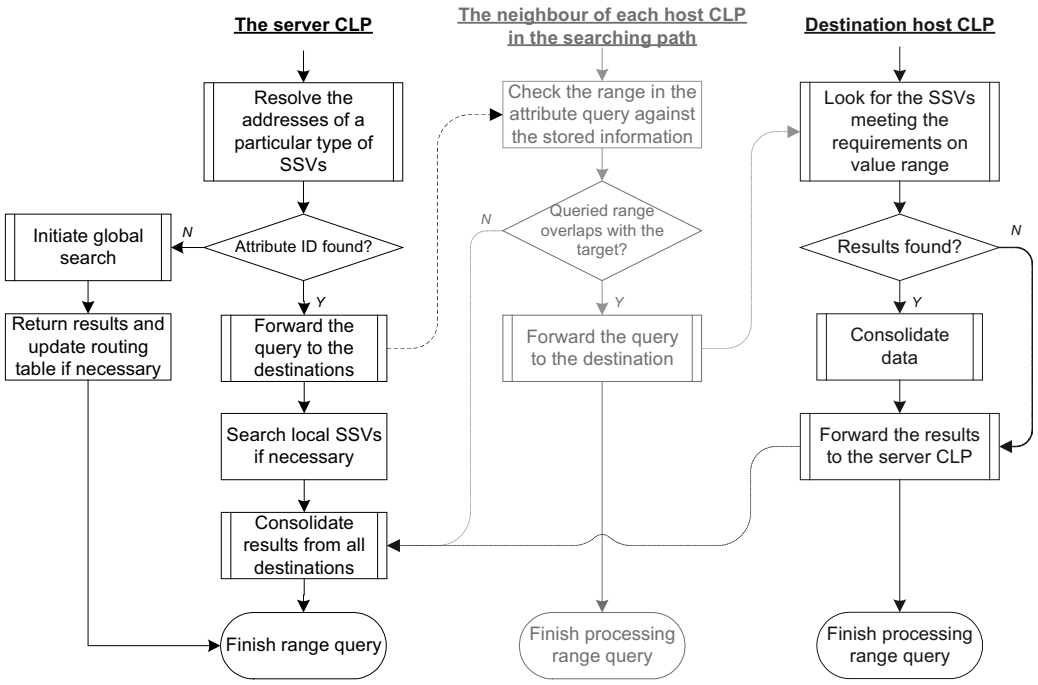

Fig. 6. An address-based routing algorithm.

from its routing table. If the attribute ID associated to the SSVs cannot be found in the table, the server CLP will initiate a global search for the SSVs with the designated type in the CLP tree. The routing table will then be updated with the returned results, and the current range query finishes. If the attribute ID can be found in the server CLP's routing table, the range query request will then be forwarded to the destinations. When the request reaches the CLP next to a destination, the CLP checks the query's range against the attribute range information about the destination to see whether both ranges overlap. If they don't overlap, search on the destination CLP will be given up. If they overlap, the query will be delivered to the destination CLP, and the results will be returned to the server CLP. Eventually, the server CLP consolidates results (if any) collected from itself and the other destination CLPs, and conveys those to the corresponding ALP, and the range query completes.

3.1.2. ID query with address-based routing

The algorithm for an ID query is straightforward. When an ALP issues an ID query, the server CLP locates the destination (if the SSV is not maintained locally) from its routing table and forwards the query to the particular host CLP.

3.1.3. Migration of shared state variables

When an SSV migrates, the change of the SSV's address immediately affects the ID-to-address binding. The routing tables in the server CLPs have to be updated with the new address. A gradual address updating scheme is used to avoid global propagation for updating addresses. This is illustrated in Fig. 7. The port through which an SSV migrates is recorded in the original host CLP, and the CLP becomes the SSV's *correspondence* CLP. The map between port and migrated SSVs is looked up as another routing table for searching those SSVs. When ALP_x originally accessed SSV_1, CLP_m was SSV_1's original host CLP. Obviously, at that time the query from ALP_x on SSV_1 must be routed to CLP_m's right port along a fixed path (the original path).

After SSV_1 migrates to a new host, from ALP_x's point of view, there are three different cases: (1) SSV_1 has been pushed further away (Fig. 7A), (2) SSV_1 has been brought closer to ALP_x (Fig. 7B) or (3) SSV_1 has been moved elsewhere (Fig. 7C). For case (1), when a new query reaches CLP_m, it will be forwarded to CLP_m's direct neighbour beyond its upper port. The forwarding will be relayed until the new host is found. For case (2), the new host must be an intermediate node in the original path, and the query will be answered straightforwardly when reaching the host. For case (3), along the original path, the query will pass a correspondence CLP_n, and then it will be relayed downwards to the new host CLP of SSV_1. From the above discussion, it is clear that no matter in which new host an SSV locates, the query will only travel along the unique path between the ALP and the new host. The routing approach does not concern any other off-path CLP.

As a result of migrating an SSV, the value range of its type may change in both the original and the new host. Therefore, the attribute value ranges related to the two hosts have to be re-computed and updated. However, this does not influence the routing on other SSVs.

3.2. Range-based routing

The range-based routing approach uses information about the attribute value range to locate SSVs in the tree. Under this approach, a CLP forwards a query according to (a) the availability of the SSV type being queried beyond its ports and (b) the value range of SSV(s) belonging to the given type.

3.2.1. Range query with range-based routing

The approach matches the query window with the attribute value ranges along the searching paths to gradually approach the potential targets. When an ALP issues an attribute value range query, a range-based routing will start from its server CLP. Searching will stop at the directions where the query window and attribute value ranges do not overlap. Like the address-based routing, the range-based routing stores the value range for each port by segmenting it. But instead of only considering the neighbour CLPs, each bit marks the existence of matching SSVs somewhere beyond the port. The port information will be kept up to date according to returned messages from neighbours; if an empty message is returned, there

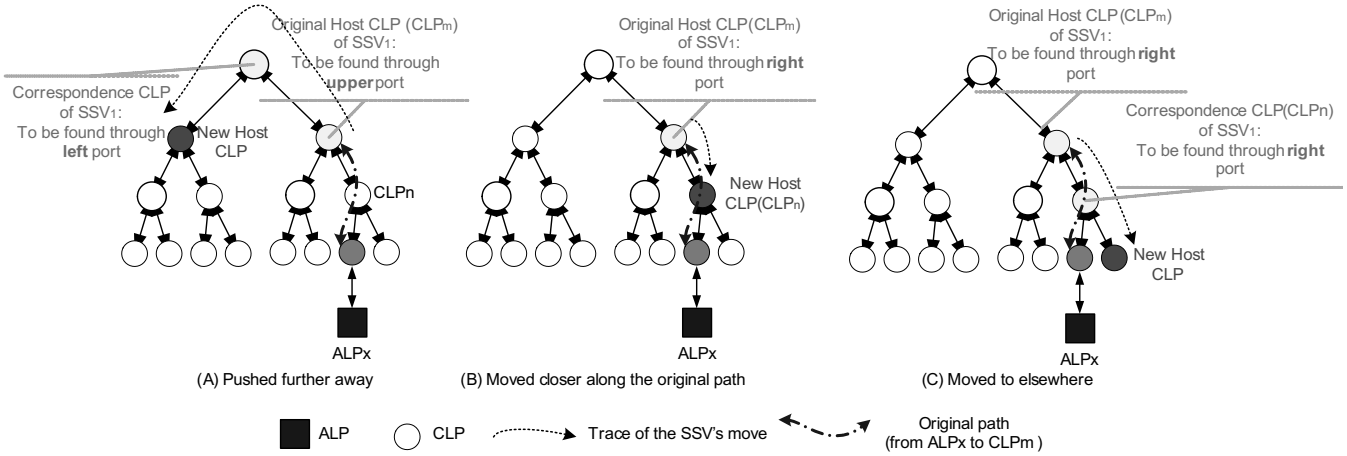

Fig. 7. Gradual address updating and routing.

are no matching SSVs behind the corresponding port and this information will be used in the future. Obviously, the port information may need to be updated when the value of any SSV changes.

In the example shown in Fig. 8, CLP_m keeps a record of the SSVs within the enclosed areas in its three ports, respectively. Suppose that a range query for SSV type "*X*-pos" and range [2,6] reaches CLP_m: The CLP has two possible directions to relay the query. Direction *B* will be given up as the attribute value range [8,9] does not overlap the window [2,6]. The query will be forwarded only towards direction *A*, as the corresponding attribute value range [1,5] matches the condition. Fig. 9 depicts the range-based routing algorithm, it applies to any intermediate CLP which receives a range query request from a port. The CLP checks whether there is any SSV matching the queried value range. After that, the attribute ranges maintained by the other two ports will be compared with the queried range, the query will then be sent to the other CLPs through the ports whose attribute values overlap the range query.

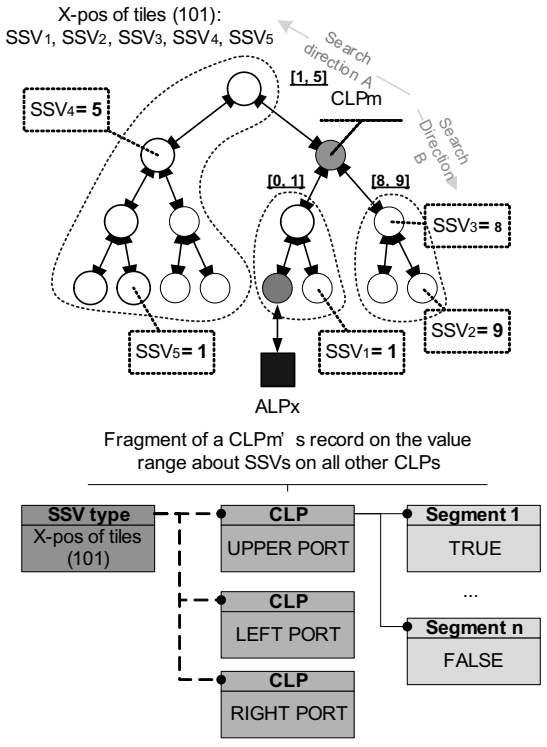

Fig. 8. Illustrating a design for range-based routing.

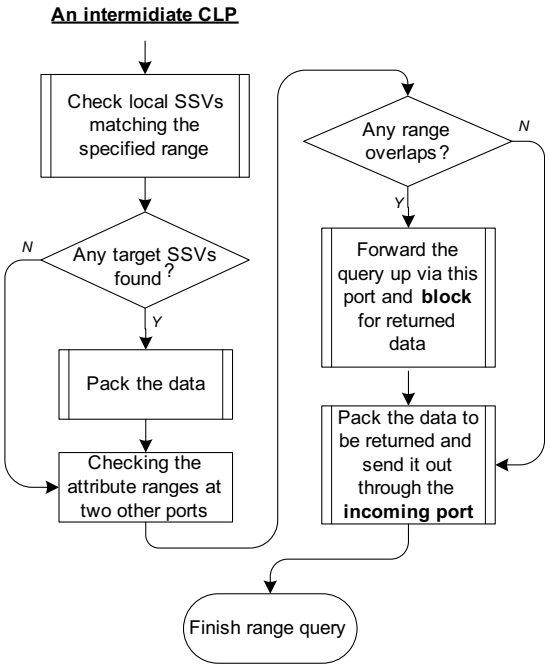

Fig. 9. A range-based routing algorithm.

The CLP waits for the results (if any) returned from the other ports and conveys them together with local targets (if any) to the port via which the range query arrives. The range query operation on this CLP completes.

3.2.2. Shared state variable (ID) query with range-based routing

The above approach can also be applied to access SSVs by ID. In this case, the ID number range is segmented as well, so that ID queries can be resolved in a similar way as Range queries. Furthermore, load management has direct influence on the ID range. After an SSV migrates, the ID range of its original and new host CLP may change. This issue is similar to dealing with the SSV update in attribute value range query.

4. A qualitative comparison

This section discusses the respective advantages and disadvantages of the two proposed approaches from a qualitative point of view.

4.1. Availability of shared state variables

Both approaches provide correct information of where SSVs may be available and avoid routing accesses to those CLPs which do not contain any SSV of the requested type. The major difference is that the address-based approach gives exact locations while the range-based approach directs the accesses to the correct searching paths.

4.2. Efficiency in performing range queries

When performing a range query, the range-based approach forwards the query to the potential targets in a bottom up fashion. In the target narrowing procedure, those out-of-range SSVs can be filtered out effectively. The address-based approach needs to route the access to the neighbours (in the searching path) of all CLPs to match the query and the host CLPs ranges. In general, the range-based approach forwards a range query to a smaller set of CLPs than the address-based solution does. The two approaches require searching within the same number of potential host CLPs. They do not differ in the complexity of searching within those hosts or the overhead in delivering results to the requesters.

4.3. Complexity for maintaining range information

The range-based approach relies on the attribute value range information. From a CLP's point of view, the attribute value ranges (in the other CLPs beyond each of its ports) must be available and accurate. A range-based algorithm needs to manage the segmented ranges properly. A fine-grained segmentation can obtain accurate routing, while a coarser segmentation will reduce broadcasting of updated ranges. In the address-based approach, no range information is maintained or broadcasted. A CLP needs to simply compute its local attribute value range prior to notifying its direct neighbours. In the case of handling SSV migration, the address-based approach does not incur any extra communication overhead for routing. In contrast, the range-based approach has to consider the immediate impact on the previous and current host CLPs, which may involve updating attribute value ranges on the CLPs.

4.4. Efficiency of ID queries

The address-based approach is able to resolve the address at the server CLP immediately for an SSV ID query, and the SSV can be accessed via a fixed path without routing to irrelevant CLPs. Using the range-based approach, querying an individual SSV is not straightforward.

4.5. Complexity for maintaining routing information

The address-based approach assigns different tasks to different CLPs (servers and intermediate nodes in the tree) with server CLPs keeping addresses of all the SSVs in its client ALPs' interest. However, address resolving within a centralised node can be optimised. The address change of any SSV does not affect routing. On the other hand, the range-based approach distributes the routing information throughout the CLP tree in an implicit manner. The address changing of any SSV may affect multiple CLPs or even the entire CLP tree.

5. Model of the simulation system

A comparative and quantitative analysis of the two proposed approaches is a non-trial task, as it involves the evaluation of efficiency in performing range queries and ID queries, the complexity of maintaining routing information, and maintaining range information, design complexity, etc. From the scale of the CLP tree and the number of SSVs, it is relatively straightforward to estimate the computational and communicational complexity of the address-based routing approach using mathematical approaches (Epstein and Axtell, 1996). However, the evaluation of range-based routing needs to consider other complicated factors at both application level and simulation level.

For a quantitative analysis, one approach would be to directly implement and integrate the two approaches into the PDES-MAS kernel. However, this would require considerable implementation efforts, and at least part of the implementation could be in vain, as the strategies may not meet the performance requirements. To avoid this we have adopted a meta-simulation approach, as proposed for instance in Liu et al. (1999), Perumalla et al. (2005), Rajive et al. (2001).

The design of the meta-simulation (Ewald, 2006) follows a layered approach similar to He et al. (2003) and Ioannidis (1996). At the top layer, we find the application model, which is responsible for generating realistic query patterns. The next layer is the middleware layer, where the routing approaches are described and the PDES-MAS framework is represented. The third layer, which typically is reserved for the model of the underlying infrastructure, is implicitly represented in the performance measurements which are integrated by calculating the costs of queries in the second layer. Thus, similar to many simulations of P2P systems, the characteristics of the underlying network are abstracted away by only counting hops and messages.

5.1. Application model

The application model focuses on the simulation of situated agents, wherein an agent has a position within the model that determines its region of interest: only objects situated in the region can be accessed by the agent. In addition, situated agents are usually able to change their own positions. This behaviour was modelled for a two-dimensional environment, as shown in Fig. 10. An agent moves step-wise towards a pre-selected target along the shortest path, and it randomly chooses a new target on arrival. The distance of the new target, the *target distance* (mark "a"), is defined by the number of steps it takes the agent to reach it. The distance an agent can move in each step is referred to as the *step size* (mark "b"), which reflects the rate of change of the agent's access pattern. The step size and target distance determine the activity scope and movement speed of an agent. After each step of movement, an agent generates ID or Range queries concerning its current region of interest.

We assume that all SSV types within the MAS model have a spatial meaning, i.e., the value ranges for Range queries reflect the actual positions of the agents. Each SSV type represents a certain dimension of the environment, such as "*X*-pos" or "*Y*-pos". Note that this assumption should not affect the generality of our model. We assume that other SSV types, such as non-spatial attributes of the modelled objects, are accessed on demand *after* range queries identified all objects in the agent's region of interest. SSVs may have a uniform value distribution or multiple normal distributions, illustrated in Fig. 11A and B, respectively.

5.2. PDES-MAS model

The model of the PDES-MAS framework has been simulated using discrete time steps, and it is formed by a set of SSVs. Each SSV consists of its (unique) ID, type, value and position in the CLP tree. The modelled CLP tree is binary and complete; therefore its structure can be defined by its depth. Another important parameter is the number of segments used by both routing algo-

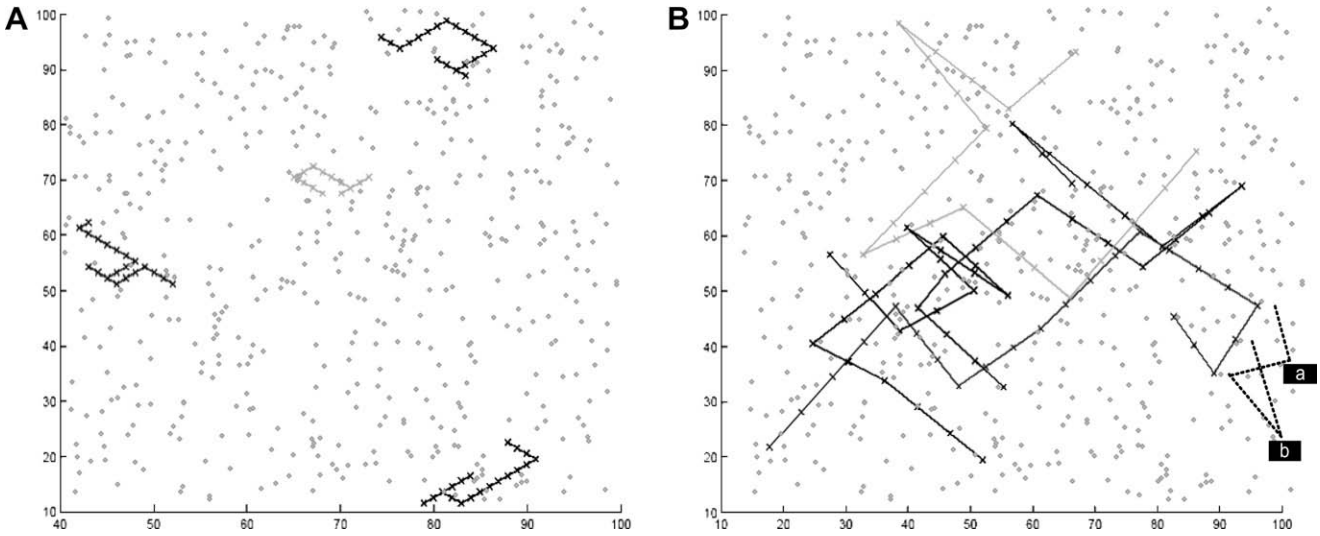

Fig. 10. Illustrating different agent movement patterns.

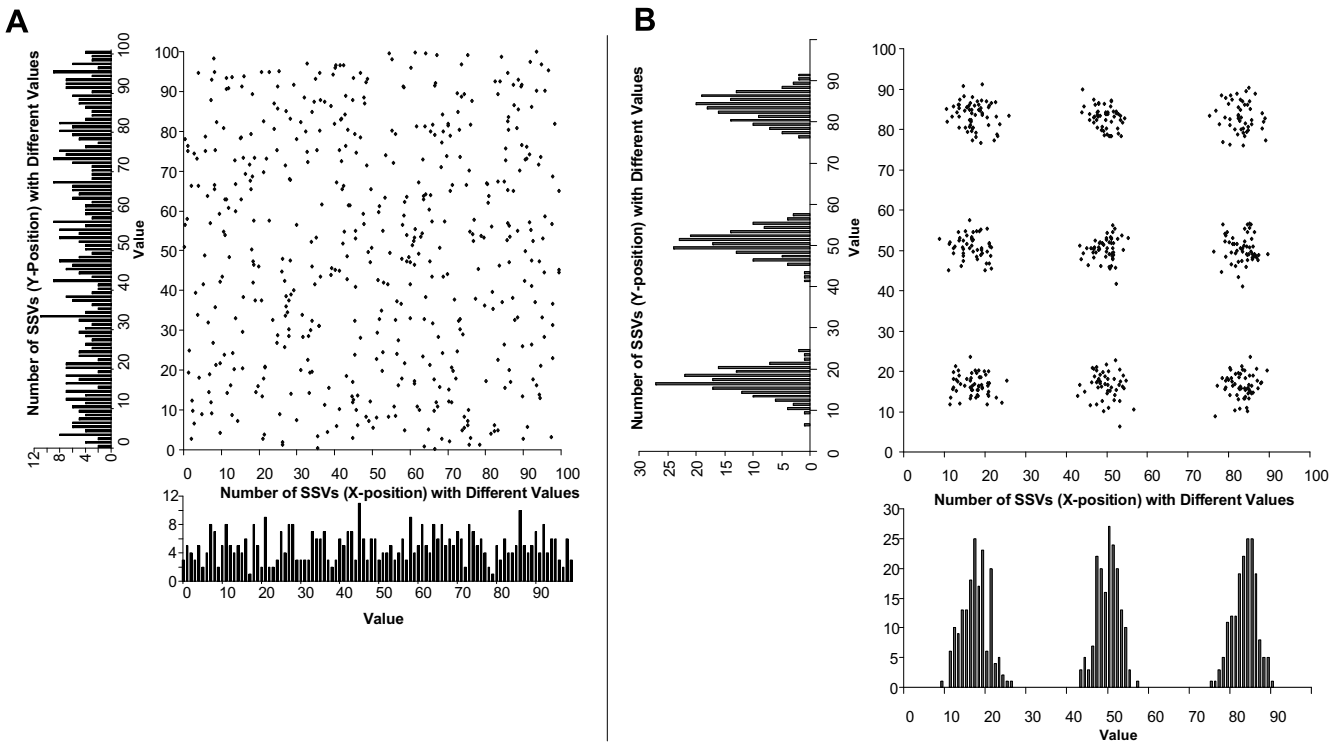

Fig. 11. Illustrating different SSV value distribution patterns.

rithms, which determines the granularity of the description of the value distribution of SSVs. To precisely identify the effect of each individual "impact factor", the model adopts different SSV distribution patterns for different runs,[5] while assuming that the locations of SSVs are fixed (as proposed in Oguara et al., 2005) in each run. This avoids bias and additional parameters to be explored, since the dynamics of load balancing might be very complex. Instead, parameterisable properties of the SSV distribution are introduced in the next section. As the SSV distribution would be controlled by any load management scheme, its impact on routing algorithm performance allows to investigate the relationship between routing and load balancing in a more general way. Nevertheless, the actual performance of both routing algorithms in combination with concrete load management schemes (such as Oguara et al., 2005) is an interesting research issue.

6. Experiments and results

The experiments aim to study the impact on the routing algorithms by (1) the distribution patterns of the values of SSVs in

[5] The overall impact of combing routing and load management will be benchmarked on the PDES-MAS framework using realistic cases.

Table 1
Summary of the default application level parameters

Name of parameters	Value
Time steps	300
Events to be generated per agent per time step	8
Step size of all agents per dimension	$\mu = 2.0$, $\sigma = 1.0$
Target distance of all agents in steps	$\mu = 5.0$, $\sigma = 2.0$
Agent's range of interest	2.0
Environment of the agents	100×100 Torus

Table 2
Summary of the default simulation level parameters

Name of parameters	Value
Depth of the CLP tree	4
Number of client ALPs to each server CLP	4 ALPs per server CLP
Root imbalance	0
CLP fluctuation	1
Number of segments for routing algorithms	100

the value space (*SSV Value Distribution*); (2) the behaviour or access pattern of ALPs on SSVs; (3) the physical distribution of SSVs in the CLP tree (*SSV Distribution Pattern*); and (4) the distribution of the values of SSVs in each individual CLP.[6] Factors 1 and 2 are at application level, related only to the agents and their environment, whereas factors 3 and 4 are at simulation level and can be parameterised. Moreover, certain additional factors have particular influence on the range-based routing approach, namely (5) the *ratio* of number of updates to number of range queries; (6) the *fluctuation* between the updated value and the original value; and (7) the *granularity of value segments*.

The environment for the experiments is set as a two-dimensional space containing 6200 objects of eight types. SSVs are defined as the *X*-pos or *Y*-pos of any object in the environment (12,400 SSVs of 16 types), and they have numerous value distributions. On initialisation, the 64 agents are distributed in a random, uniform manner over the space. The default settings of other parameters for the experiments are summarised in Tables 1 and 2. The step size per dimension and the target distance conform to normal distributions, ($\mu = 2.0$, $\sigma = 1.0$) and ($\mu = 5.0$ and $\sigma = 2.0$), respectively. The diameter of an agent's region of interest is set to 2.0. For example, an agent at the Pos-*X* = 50 could query a range of [49, 51] for all types associated with this dimension. The parameters roughly represent average values between worst and best cases for each routing approach. This allows us to observe the behaviour of each algorithm in a relatively small parameter space. The parameter space region that we investigated contains both kinds of setups: those in which the address-based routing is preferable, and those in which range-based routing is better (as illustrated for instance in Figs. 10, 11 and 21). In each of the following experiments, we always adjust one parameter while applying default values to the rest. This tends to explore the "general" scenarios in reality, and avoids biased analysis due to only using special cases with extreme settings. Each agent randomly generates eight requests for any type of SSV in each step. The simulation executes 300 time steps, which allows the simulated system to evolve significantly.

The physical distribution of SSVs can be adjusted with two parameters, namely *root imbalance* and *CLP fluctuation*. Root imbalance describes the percentage of additional SSVs hosted by the root CLP comparing to the rest. When root imbalance = 0, SSVs are distributed evenly on all CLPs. When root imbalance = 1, all SSVs are concentrated on the root CLP. Let n be the number of CLPs, $r \in [0, 1]$ the root imbalance, and $s_i \in [0, 1]$ the share of SSVs hosted by CLP_i, i.e., $s_2 = 0.6$ means that 60% of all SSVs are hosted by CLP_2. We can then define s_1 (the share of SSVs on the root CLP) as $\frac{1+r\cdot(n-1)}{n}$ and s_i with $i = 2, \ldots, n$ as $\frac{1-r}{n}$, so that $\sum_{i=1}^{n} s_i = \frac{1+r\cdot(n-1)}{n} + (n-1)\cdot\frac{1-r}{n} = 1$. The CLP fluctuation constraints the maximum difference between the greatest and smallest value of SSVs of the same type on an individual CLP. For example, suppose CLP fluctuation = 0.05 and the designated type of SSVs has a value space from 0 to 100, then the difference of these SSVs' value on the CLP is not greater than $0.05 \times 100 = 5$: A CLP may host two SSVs of type *X*-pos with values 80 and 82, but another SSV of type *X*-pos, with value 75, cannot be hosted by the same CLP, because $82 - 75 > 5$. Since SSVs do not migrate but have dynamic values, this condition holds only for the initial state. The experimental results are reported in terms of routing cost and accuracy.

The routing cost is measured using two metrics: the *number of messages* and the *number of hops to be traversed* in resolving each access to SSVs. The number of messages is the number of all messages that are generated by the routing algorithm in order to resolve a query. We regard the transmission of information from one CLP to another as a single message. If the same information is propagated along several CLPs, it is counted as multiple messages. Hence, the number of messages is a measurement of the overall bandwidth consumption. The number of hops is the maximum number of messages that had to be sent sequentially until the request could be resolved. This means, that the number of hops corresponds to the maximal path length from the ALP generating the request to a CLP which had to be contacted, multiplied by 2 (for the query and the corresponding response). This is very similar to the notion of critical paths in parallel computing. Hence, the number of hops is a measurement of the overall latency. The two metrics for the routing algorithms are denoted by the variables $\text{range-based}_{\text{messages}}$, $\text{range-based}_{\text{hops}}$ and $\text{address-based}_{\text{messages}}$, $\text{address-based}_{\text{hops}}$ for the range-based and address-based algorithm, respectively.

In order to calculate the accuracy of the routing algorithms, the minimal number of messages and hops ($\text{opt}_{\text{messages}}$ and opt_{hops}) are computed, which constitute the absolute limit for optimising the cost of any routing algorithm. For instance, Fig. 1 illustrates the querying of two SSVs, with the smallest set of CLPs and connections for this query highlighted. Each connection needs to transmit two messages (one request and one response), thus the total number of messages is 10. The maximum number of hops is recorded as eight (for reading SSV_1 on the left); this is because messages have to be sent sequentially (in any path) until reaching the target and thus the latencies in other concurrent search paths are masked. The communication cost between the ALP and its server CLP is considered negligible. The ratio of the minimal number of messages (or hops) to the number of messages (or hops) is defined as the *accuracy* of routing algorithms. For example, the "message accuracy" of the range-based algorithm is defined as $\text{accuracy}_{\text{messages}}^{\text{range-based}} = \frac{\text{opt}_{\text{messages}}}{\text{range-based}_{\text{messages}}}$. Likewise, the "hop accuracy" for ranged-based algorithm and the message and hop accuracy of the address-based algorithm are denoted by $\text{accuracy}_{\text{hops}}^{\text{range-based}}$, $\text{accuracy}_{\text{message}}^{\text{address-based}}$ and $\text{accuracy}_{\text{hops}}^{\text{address-based}}$, respectively.

6.1. Effect of agent's environment (SSV value distribution)

Several non-uniform distributions of SSV values have been used while keeping default values for all other parameters. Values have been assigned to SSVs in a round-robin fashion by one of three normal distributions (see Fig. 11B). The mean values of normal distributions are $16\frac{2}{3}$, 50 and $83\frac{1}{3}$). The deviation σ is varied from 0 to 10, so that the SSV value distribution changes from highly concentrated to highly scattered.

[6] This is different to the aforementioned value distribution parameter. For example, in a scenario the value ranges of SSVs in all CLPs are very close, and in another scenario the value ranges are significantly distinct from one CLP to another.

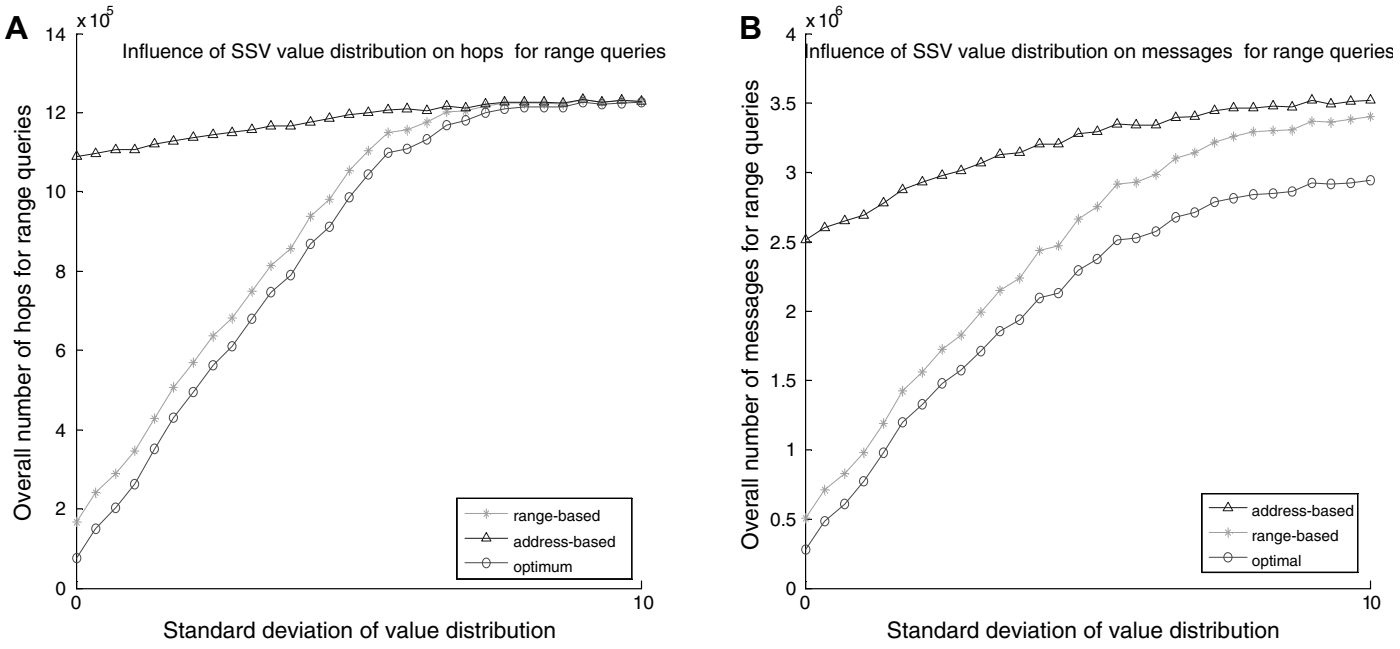

Fig. 12. Influence of overall SSV value distribution on routing cost.

The effects of value distribution on the cost and accuracy of routing algorithms as compared to the optimal cost are reported in Figs. 12 and 13, respectively. Evidently, with values of SSVs distributed more sparsely, routing becomes more costly. In terms of number of hops, the range-based algorithm performs much better, by closely approaching opt_{hops}. When SSV values are scattered enough, the number of hops in both algorithms converges to opt_{hops}. In terms of number of messages, the performance of both algorithms deteriorates as the degree of dispersion of SSVs increases. The $accuracy_{hops}$ of both algorithms converge to 1 while the maximum $accuracy_{messages}$ for both algorithms only approximates 0.8. In all situations, the range-based algorithm is likely to incur less overhead for routing range queries.

Similar experiments have also been performed to measure the accuracy of making ID queries. The address-based algorithm always achieves optimal results (accuracy = 1) while the accuracy of the range-based algorithm is quite low, about 0.36. This is because the range-based algorithm does not store the location of an individual SSV and requires exhaustive range searches on SSV IDs.

6.2. Effect of SSV value distribution pattern

A set of experiments have been performed to examine how the distribution pattern of SSVs in the simulation infrastructure (the CLP tree) affects the performance of routing algorithms. This subsection reports the effect of two simulation level factors: (a) the

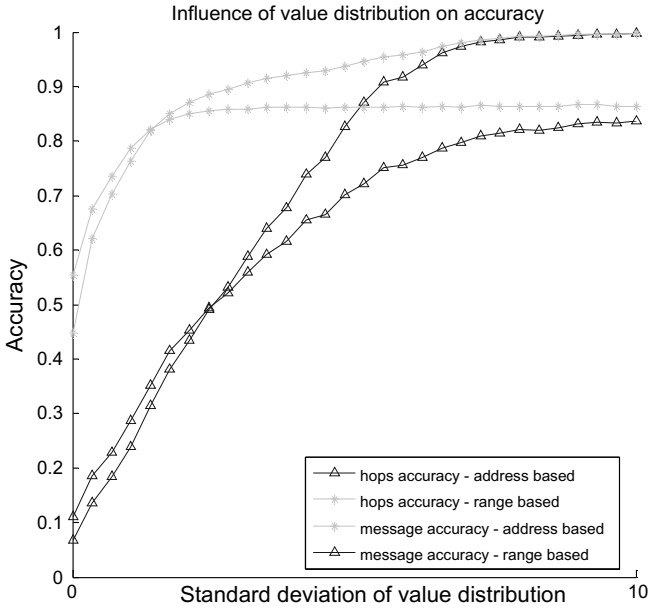

Fig. 13. Influence of overall SSV value distribution on accuracy of routing.

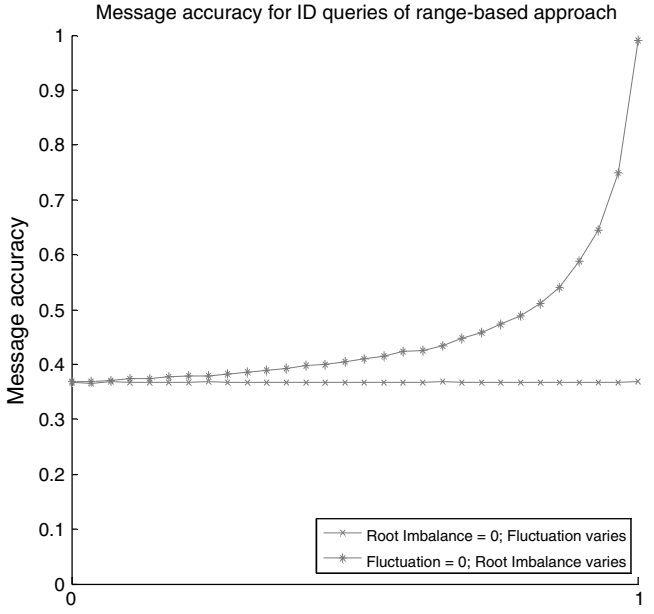

Fig. 14. Message accuracy for ID query.

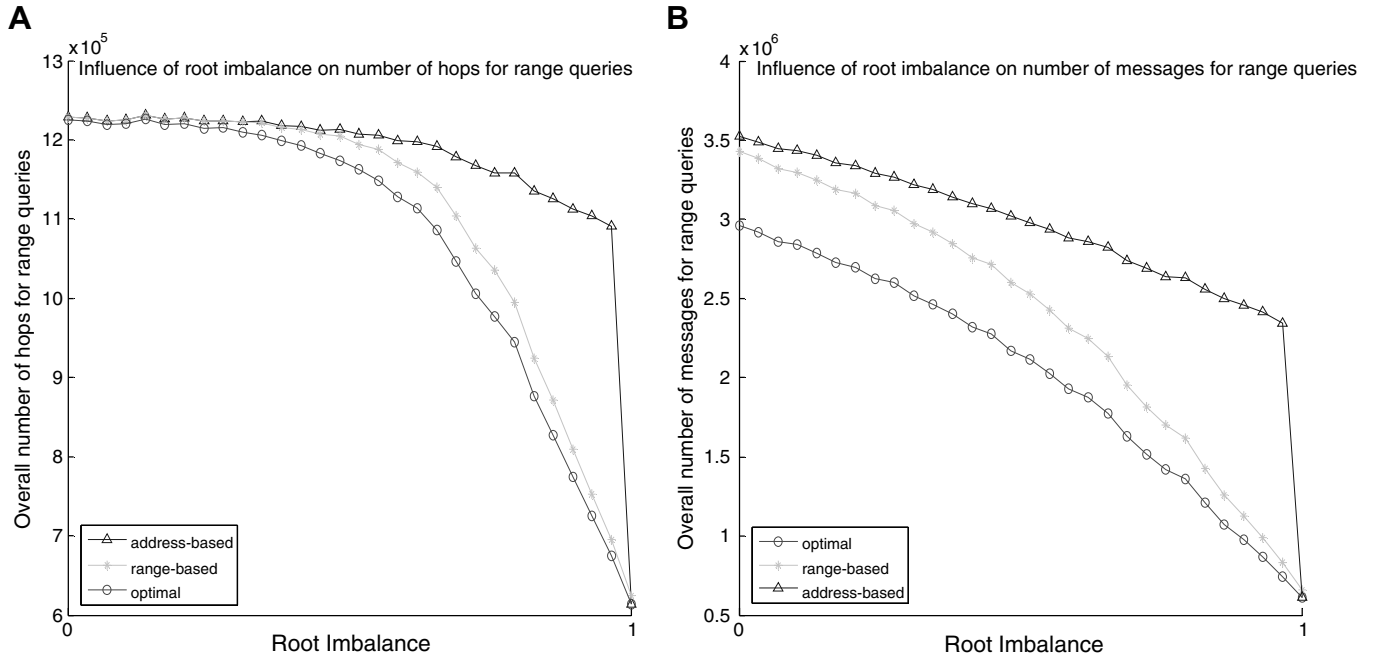

Fig. 15. Influence of the SSV distribution pattern on the routing cost.

physical distribution of SSVs on the CLP tree, and (b) the distribution of SSVs' values on each individual CLP, given the behaviour of agents remains the same. The two relevant parameters, *root imbalance* and CLP *fluctuation*, are varied between 0 and 1 as well as 0.05 and 1, respectively.

Fig. 14 gives the accuracy of the range-based algorithm for ID queries in terms of messages. The SSV value distribution in a CLP does not affect the routing of ID queries at all. However, concentration of SSVs on fewer CLPs will dramatically improve the accuracy of routing ID queries using range-based algorithm.

Fig. 15 illustrates the impact of the SSVs' physical distribution pattern on the cost of routing range queries using the two routing algorithms. When SSVs are distributed on all CLPs uniformly (root imbalance = 0), range-based$_{hops}$ and address-based$_{hops}$ are nearly equal to opt$_{hops}$ while range-based$_{messages}$ and address-based$_{messages}$ are very close but still much greater than opt$_{messages}$. With the increase of root imbalance, the routing cost gradually decreases. The range-based algorithm adapts much better than the address-based algorithm. When root imbalance = 1, all SSVs locate at the root CLP and it makes no difference to either algorithm; this extreme case reflects the use of a single centralised CLP.

Fig. 16 presents the cost of performing range queries against the value distribution on each CLP. The results are similar to those obtained in Fig. 12, namely the value distribution of all SSVs and the value distribution of SSVs on each individual CLP both have significant influence on the routing cost involved in range queries.

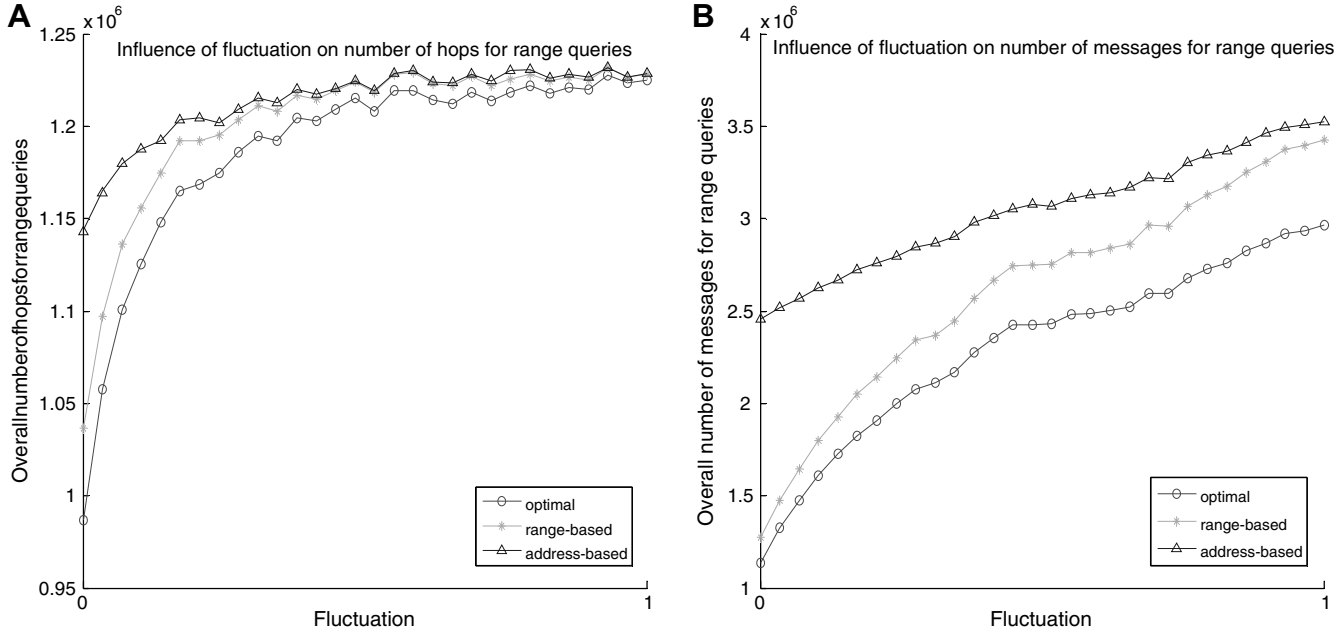

Fig. 16. Influence of SSV value distribution on individual CLPS on routing cost.

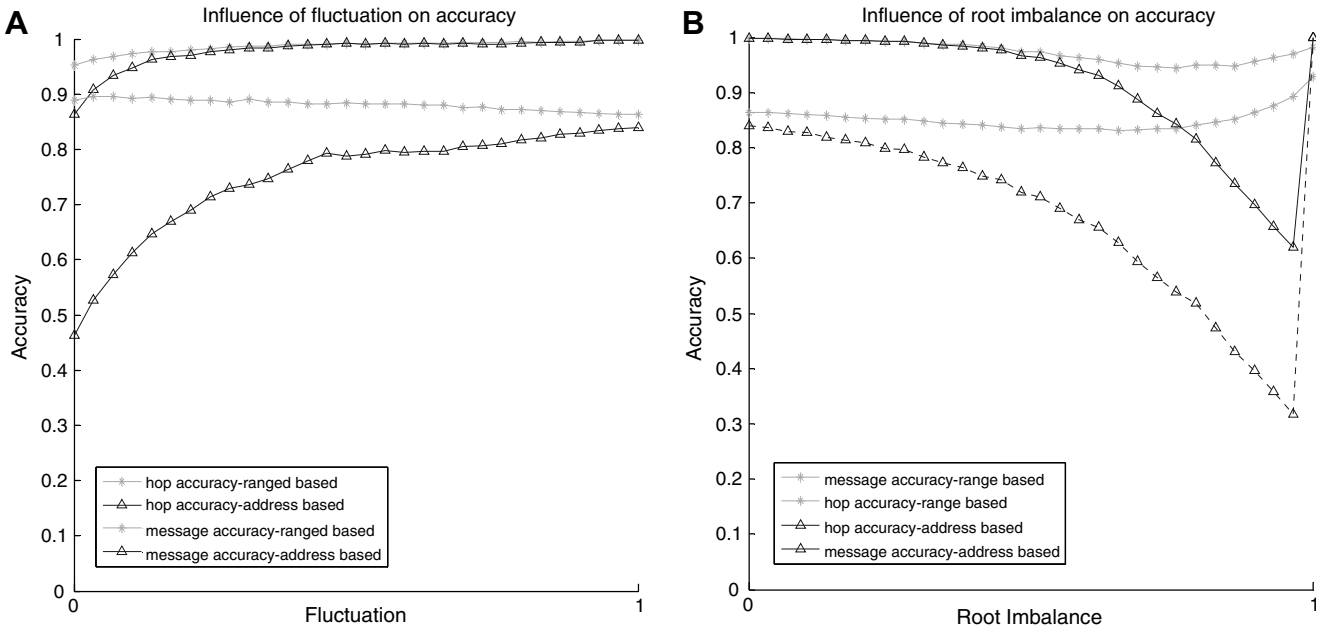

Fig. 17. Influence of SSV's physical distribution pattern on the accuracy of routing algorithms.

The accuracy of the routing algorithms against the SSVs' physical distribution pattern is illustrated in Fig. 17. These results further indicate that the range-based algorithm adapts better to the SSV's distribution pattern. The latencies (number of hops) incurred by both algorithms are similar; however the range-based algorithm generates much less communication traffic (number of messages). Furthermore, comparing to Fig. 13, the value distribution on an individual CLP has a less significant impact than the overall value distribution.

6.3. Further analysis of the range-based approach

The complexity of the range-based approach calls for further investigation. Three additional key parameters may affect the performance of range-based routing: (a) the *ratio* of the number of updates to number of range queries, (b) the *fluctuation* between the updated value to the original value, and (c) the *granularity of segments*.

6.3.1. Effect of SSV update (application level)

An agent may randomly update any SSV whose value is in its region of interest. In the meta-simulation, the SSV's value is updated using an absolute offset (to the old value), which is randomly set conforming to uniform distribution with lower bound 0 and upper bound varied between 0.01 and 5. Furthermore, the probability that an agent generates an update query is set between 0.01 and 0.5.

Fig. 18 gives an overall picture of the correlation between offsets of updated SSV values and the ratio of update queries to the messages (updates) needed for accomplishing update queries.

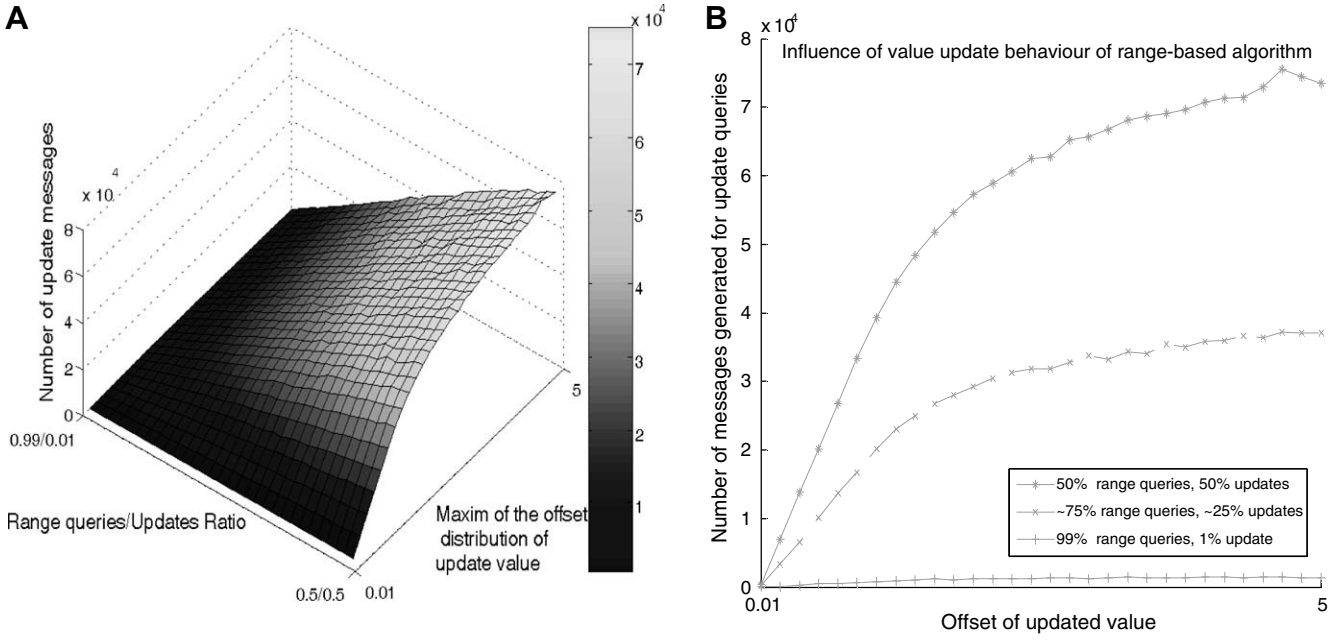

Fig. 18. Agent behaviour's influence on number of messages for update query.

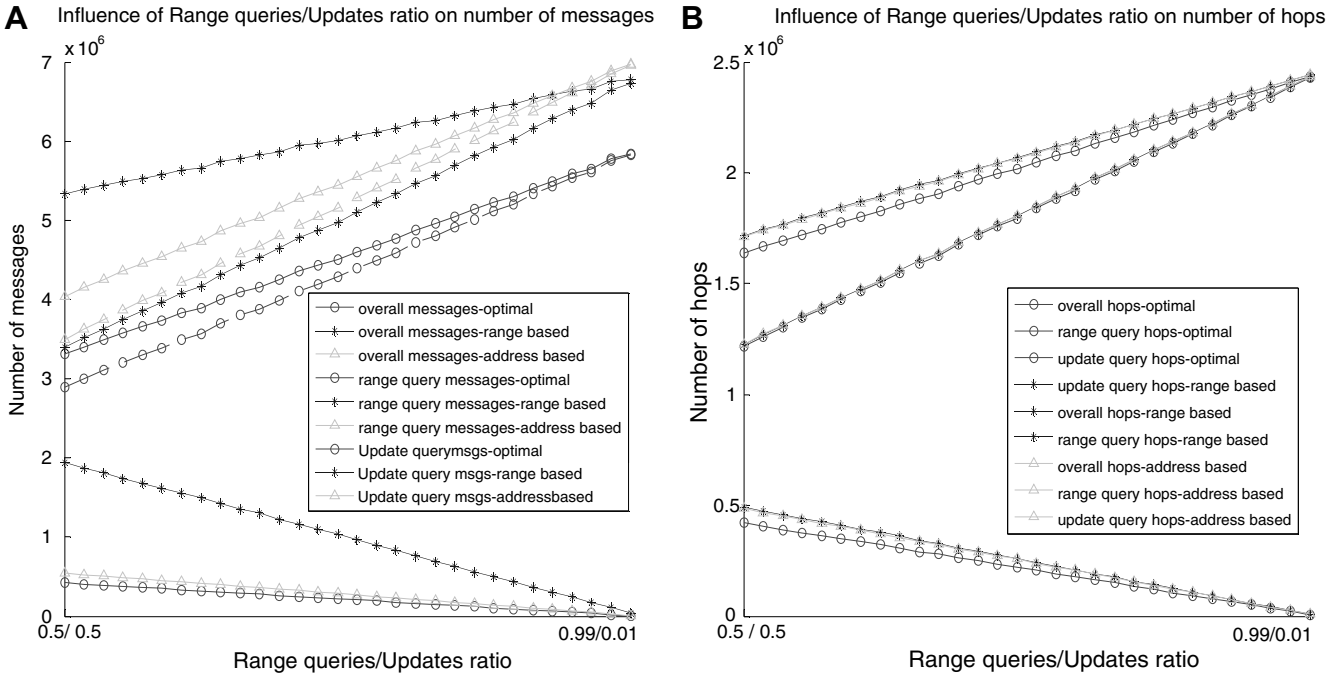

Fig. 19. Influence on the ratio of update queries to routing cost.

Three specific ratios of update queries are highlighted. The results indicate that the increase of both the offset and the ratio of update queries lead to the generation of more update messages. Compared to the number of overall messages obtained in the previous experiments (see Figs. 12,15,16), the number of update messages is very small. Although this number depends on the specific configuration of the experiments, it still suggests that the overhead incurred by update queries tends to be negligible in an environment with heavy load and traffic.

Fig. 19 illustrates the effect of the ratio of update queries to the routing cost and the breakdown of routing cost involved in update queries and range queries. This ratio has a significant impact to the number of messages and hops required by both query types. Additionally, Fig. 19A shows that basically the address-based algorithm outperforms the range-based one in situations with frequent update queries. Although the address-based approach generates more messages for range queries, it still incurs fewer messages in total due to the benefit in update queries. Fig. 19B indicates that the number of hops is linear to the ratio of update queries and the candidate routing algorithms do not make considerable difference in this aspect.

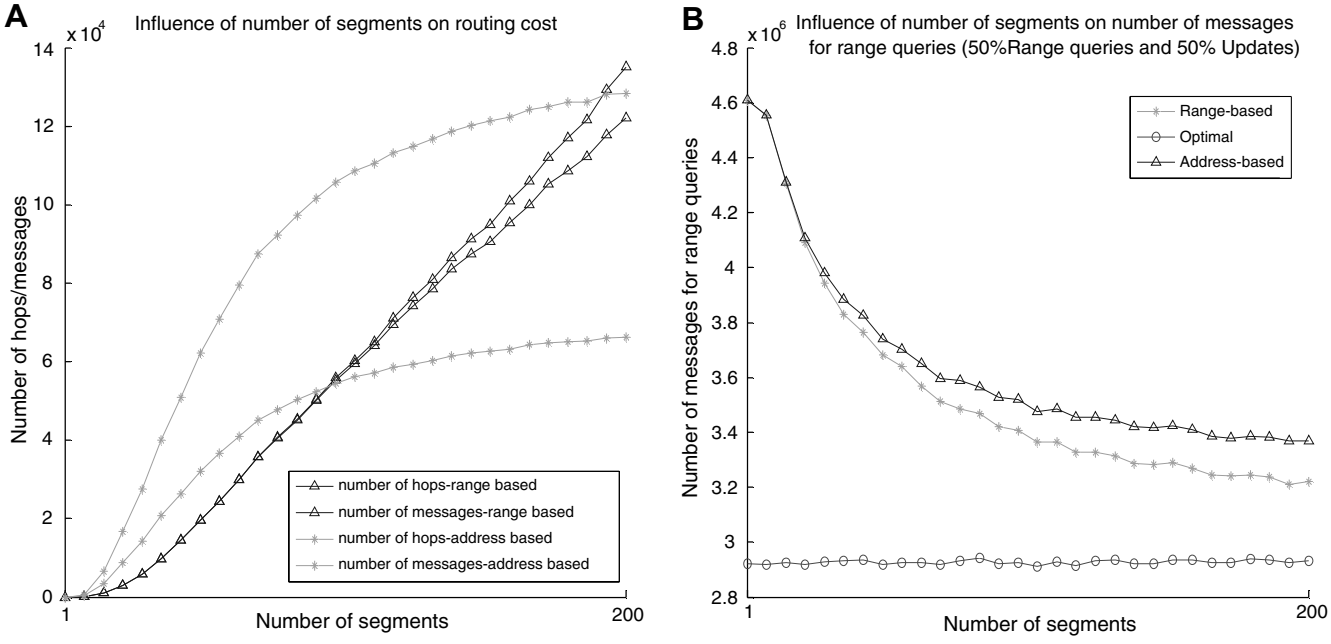

Fig. 20. Influence of segment's granularity on the routing cost.

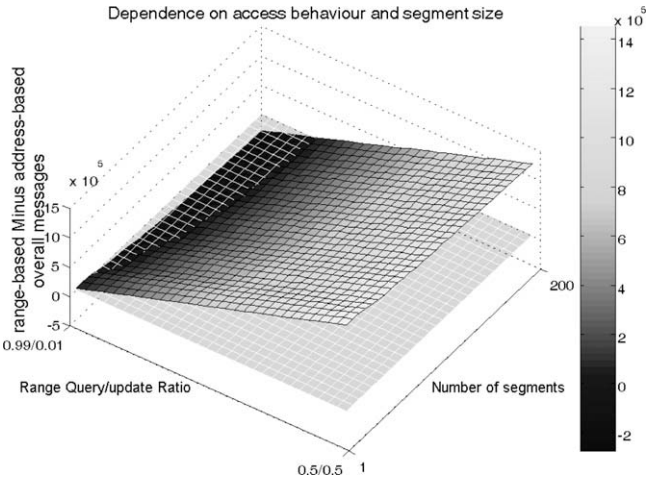

Fig. 21. Difference in message numbers using different routing algorithms.

6.3.2. Effect of granularity of segments (simulation level)

A set of experiments have investigated the role of the granularity of attribute value range segmentation for the range-based algorithm. The number of segments used to describe an attribute value range varies between 1 and 200, with a larger number implying a more precise attribute value range description.

The results are reported in Fig. 20. The total (for both range queries and updates) number of messages required by the address-based algorithm is larger in most cases. As the number of segments increases the address-based algorithm latency stabilises, as queries have to be sent only to neighbours. When the number of segments becomes large enough, the range-based algorithm tends to update CLPs in a rather wide extent, an action that deteriorates its performance. Fig. 20B illustrates the relationship of routing range queries to the granularity of segmentation. As the segment size decreases, the number of messages required to satisfy the range query is dramatically reduced, because the accurate description of the value ranges results in a more precise routing.

Fig. 21 presents the overall difference (range-based − address-based) of the two algorithms in terms of (total) number of messages. Negative values denote the space for achieving better performance for adopting the range-based approach, and vice versa for adopting the address-based one.

6.3.3. Effect of agent's behaviour

In the experiments presented in the previous sections, the environment is heavily populated (64 agents randomly situated in a 100×100 surface and 12,400 SSVs distributed either sparsely or concentrated in nine central points as shown in Fig. 11), a feature that renders the investigation of the effect of agents' behaviour a difficult task as, irrespective of how far agents move (regionally or globally), there are always plenty of SSVs for them to access in their region of interest. For this investigation, a thinly populated environment with four agents and 1550 SSVs[7] of two types has been used; the size of CLP tree has been reduced accordingly (depth = 2). In this set of experiments, an agent's behaviour pattern can be characterised by its activity scope and the speed at which it moves. For each agent, its range of interest is 5.0 and its step size is varied between 0.1 (agent moves regionally) to 5.0 (agent moves globally). Two different target distances, (1, 0) and (5, 2), are tested to mimic different movement speeds. The rest of the parameters have been set to the default values.

[7] This number represents 1/16th of the number of SSVs (12,400) in the original experiments, to reflect the number of agents used (4 = 64/16).

Fig. 22 reports the effect of the size of an agent's activity territory in terms of number of hops and messages. Fig. 22A and B reports results for "fast" agents and C and D for "slow" agents. Evidently, the address-based approach incurs more latency while its overall cost is almost neural to the agent's moving pattern. When an agent moves more globally and faster, the range-based algorithm becomes more costly as the agent generates queries that differ very much from each other, so that many updates occur.

7. Related work

The distributed simulation of agent-based systems has been the focus of several recent papers including (Anderson, 2000; Lees et al., 2007; Minson and Theodoropoulos, 2004; Minson and Theodoropoulos, 2008; Wang et al., 2003; Himmelspach et al., 2007; Riley, 2003; Riley and Riley, 2003), but none of this work has considered the efficient distribution and access of the simulation's shared state space.

Routing has been heavily studied in the area of computer networks (Tanenbaum, 2003), however the issue of range queries does not arise in this context. Steen et al. (1998) proposed a scalable location service mechanism for locating mobile objects in distributed environments. Their approach is address-based, which binds an object's name to one or more addresses where the object can be contacted. The location service is designed to handle objects with arbitrary migration patterns. Using a hierarchical search tree, where each new object is registered, the leaf nodes store the addresses. By registering the contact address in the smallest region in which the object is moving, the approach only requires searching an extremely short path to locate randomly migrating objects.

The issue of range queries has received considerable attention in the area of distributed spatial databases (e.g., An et al., 2003; Faloutsos and Kamel, 1994; He et al., 2003; Ioannidis, 1996; Ndiaye et al., 2003; Pagel et al., 1993; Samet, 1990) although the work in this area tends to focus on the optimisation of matching algorithms. In the area of context-aware computing, location dependent information services (LDIS) address the problem of mobile query sources, however, typically they do not have to cope with highly dynamic data distributions, because the data can be distributed according to its geographical validity in a static fashion (Lee et al., 2002).

The problem of coordinated access to shared data has also been studied in the context of cache consistency protocols for distributed and multiprocessor systems (e.g., Morin and Puaut, 1997; Nagaraj, 2004; Stenström, 1990), but such systems are based on data replication and therefore the solutions devised are not directly applicable to the PDES-MAS architecture.

In the modelling and simulation community, the majority of approaches focus on the use of multicast rather than unicast peer-to-peer (P2P) overlays as described here. Notable contributions by Rak et al. (1997), Boukerche et al. (2000) and Berrached et al. (1998) all used variations on the theme of mapping a set of multicast groups on to the cells of an n-dimensional grid. Range queries here are mapped to some subset of the cells (groups) to which the querying node subscribes. Alternative mappings such as that proposed by Morse et al. (1999) reduce redundant traffic, but increase the computational complexity.

The problem of how to optimally distribute and retrieve both data (publications) and queries over the data (subscriptions) in a P2P overlay has been studied extensively in the field of content addressable networks (CANs) such as the distributed hash table (DHT) (Ratnasamy et al., 2001). Approaches concentrating on serving contiguous range queries in several dimensions with minimum latency have been presented by both Barambe et al. (2004, 2006) and Ganesan et al. (2004). These systems are designed for

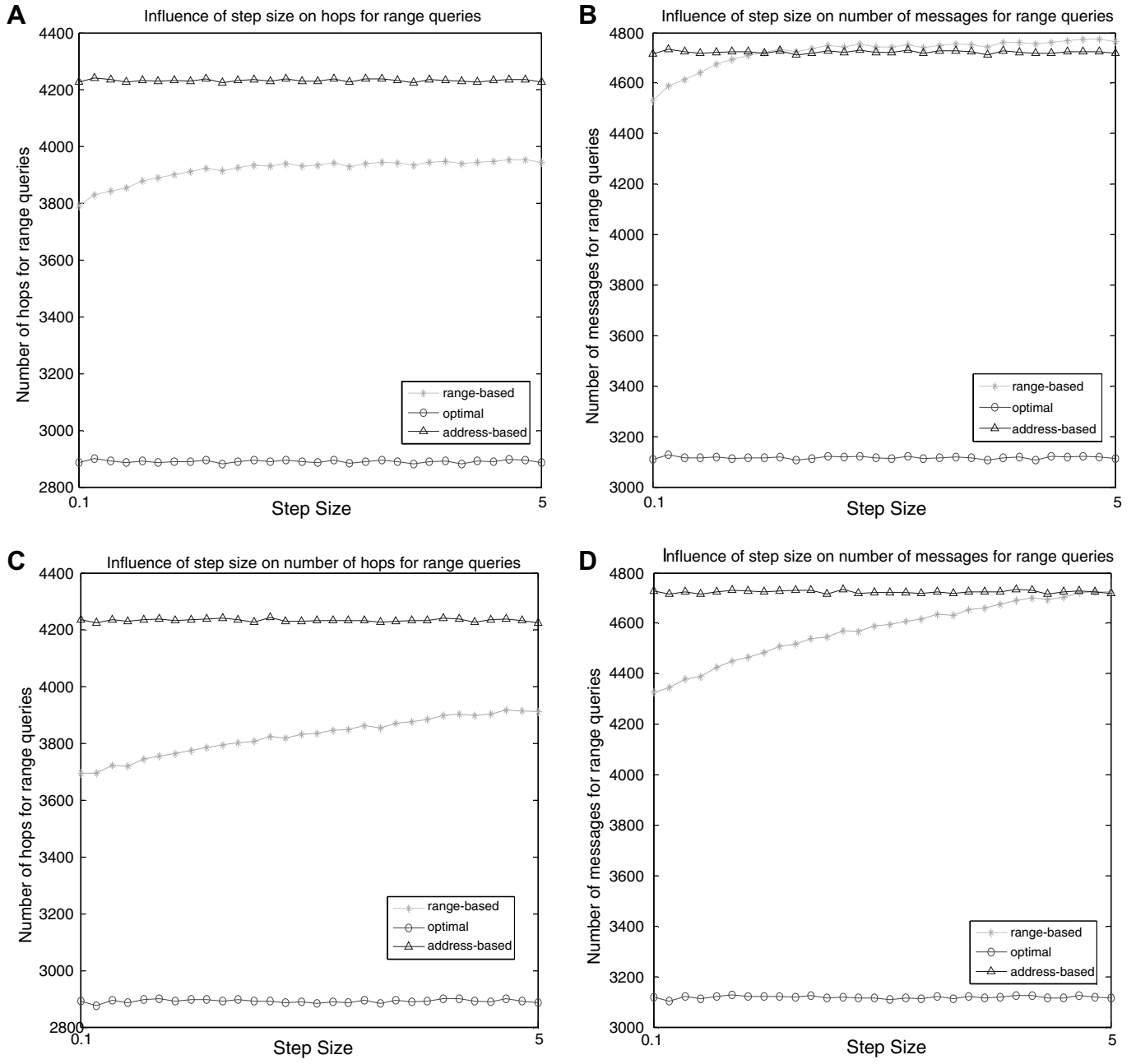

Fig. 22. Effect of agent's access pattern.

real time applications, such as P2P network multiplayer games, rather than asynchronous logical time systems like parallel simulations.

8. Conclusions and further work

This paper has identified the efficient data accessing as a key issue to optimising the execution of distributed simulations of complex agent-based systems. It has presented two different approaches to address this problem, namely an address-based approach, which locates data according to their address information, and a range-based approach, whose operation is based on looking up attribute value range information along the paths to the destinations.

To facilitate the analysis of the proposed approaches, the paper adopted a meta-simulation framework to analyse query performance on distributed and dynamic data whose location and properties are changing constantly and unpredictably. This is an open challenging problem and particularly difficult to tackle using pure deterministic or analytic approaches. Although the proposed algorithms were analysed in the context of the PDES-MAS framework, the results obtained are valid for any system with similar characteristics, such as distributed systems based on peer servers. The main conclusions that can be drawn from the experimental results obtained can be summarised as follows:

- The data value distribution has a vast impact on the performance of both approaches. When the data values are distributed sparsely enough, both algorithms can minimize the latency of routing to the optimal case, whereas they tend to generate a large number of messages otherwise (~20% more than the optimal case).

- The data distribution pattern is also a decisive factor to the performance of routing. The more evenly data is distributed, the closer both approaches approximate the optimal case. The smaller the fluctuation of data values maintained in a particular node, the more accurate the routing.
- Granularity of segmentation can considerably affect the routing of update queries. The address-based approach is superior to the range-based one when segments are very precise. In range-based routing, the routing accuracy is adapted. However, a precise segmentation leads to large overhead in dealing with update queries.

The meta-simulation analysis approach described in this paper has proved to be a powerful tool to analyse the impact of the model's abstract characteristics to the efficiency of the proposed algorithms. However, as in every simulation exercise, the problem of verification and validation of the simulation and the reliability of the results remains. Most importantly, as any meta-simulation approach, it provides only an idealistic performance projection of the algorithms and does not take into account the specific characteristics of any particular implementation or agent-based model. A more realistic performance evaluation of the proposed routing algorithms and an analysis of their scalability and their computational complexity requires their implementation and integration in the PDES-MAS kernel and experimentation with more realistic benchmarks. Work in this direction has already commenced.

Another challenging issue to be addressed in the future is the relationship between the proposed routing algorithms and the load management (data migration) and synchronization mechanisms of the PDES-MAS kernel (described in Oguara et al. (2005) and Lees et al. (2003, 2005), respectively).

Acknowledgement

This work was supported by the EPSRC under Grant No. GR/R45338/0.

References

Anderson, J., 2000. A generic distributed simulation system for intelligent agent design and evaluation. In: Proceedings of AI, Simulation and Planning in High Autonomy, Systems.

An, N., Jin, J., Sivasubramaniam, A., 2003. Toward an accurate analysis of range queries on spatial data. IEEE Transactions on Knowledge and Data Engineering 15 (2), 305–323.

Barambe, A., Agrawal, M., Shesan, S., 2004. Mercury: supporting scalable multi-attribute range queries. In: Proceedings of the ACM SIGCOMM.

Barambe, A., Pang, J., Shesan, S., 2006. A distributed architecture for multiplayer games. Technical Report, Carnegie-Mellon University, CMU-CS-05-112.

Berrached, A., Beheshti, M., Sirisaengtaksin, O., de Korvin, A., 1998. Approaches to multicast group allocation in HLA data distribution management. In: Proceedings of the 1998 Spring Simulation Interoperability Workshop.

Boukerche, A., Roy, A.J., Thomas, N., 2000. Dynamic grid-based multicast group assignment in data distribution management. In: Proceedings of the Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications.

Epstein, J.M., Axtell, R.L., 1996. Growing Artificial Societies: Social Science from the Bottom Up. Brookings Institution Press.

Ewald, R., 2006. Simulation of load balancing algorithms for discrete event simulations. Master Dissertation, School of Computer Science, University of Rostock, Germany.

Ewald, R., Chen, D., Theodoropoulos, G., Lees, M., Logan, B., Oguara, T., Uhrmacher, A.M., 2006. Performance analysis on shared data accessing algorithms for distributed simulation of multi agent systems. Presented at the 20th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2006), May 23–26, Raffles Hotel, Singapore.

Faloutsos, C., Kamel, I., 1994. Beyond uniformity and independence: analysis of R-trees using the concept of fractal dimension. In: Proceedings of the thirteenth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, June 1994. pp. 4–13.

Fujimoto, R.M., 2000. Parallel and distributed simulation systems. Wiley Series on Parallel and Distributed Computing. Wiley-Interscience, ISBN 978-0-471-18383-9.

Ganesan, P., Yang, B., Garcia-Molina, H., 2004. One torus to rule them all: multi-dimensional queries in peer-to-peer systems. In: Proceedings of the 7th International Workshop on the Web and Databases.

He, Q., Ammar, M.H., Riley, G., Raj, H., Fujimoto, R., 2003. Mapping peer behavior to packet level details: a framework for packet-level simulation of peer-to-peer systems. In: Proceedings of the 11th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2003.

Himmelspach, J., Ewald, R., Leye, S., Uhrmacher, A.M., 2007. Parallel and distributed simulation of parallel DEVS models. In: Proceedings of the SpringSim'07, DEVS Integrative M&S Symposium. SCS, pp. 249–256.

Ioannidis, Y., 1996. Query optimization. ACM Computing Surveys. Symposium Issue on the 50th Anniversary of ACM 28 (1), 121–123.

Jennings, N.R., Wooldridge, M., 1998. Applications of intelligent agents. In: Jennings, N.R., Wooldridge, M. (Eds.), Agent Technology: Foundations, Markets, Applications. Springer, New York, pp. 3–28.

Liu, J., Nicol, D., Premore, B., Poplawski, A., 1999. Performance prediction of a parallel simulator. In: Proceedings of the 13th Workshop on Parallel and Distributed Simulation (PADS'99), pp. 156–164.

Lee, D.L., Xu, J., Zheng, B., Lee, W., 2002. Data management in location-dependent information services. IEEE Pervasive Computing 1 (3), 65–72.

Lees, M., Logan, B., Theodoropoulos, G., 2003. Adaptive optimistic synchronisation for multi-agent simulation. In: Al-Dabass, D. (Ed.), Proceedings of the 17th European Simulation Multiconference, pp. 77–82.

Lees, M., Logan, B., Chen, D., Oguara, T., Theodoropoulos, G.K., 2005. Decision-theoretic throttling for optimistic simulations of multi-agent systems. In: Proceedings of the Ninth IEEE International Workshop on Distributed Simulation and Real-Time Applications, October 2005. pp. 179–186.

Lees, M., Logan, B., Theodoropoulos, G.K., 2007. Distributed simulation of agent-based systems in HLA. ACM Transactions on Modelling and Computer Simulation 17 (3), ISSN: 1049-3301. <http://doi.acm.org/10.1145/1243991.1243992>.

Logan, B., Theodoropoulos, G.K., 2001. The distributed simulation of multi-agent systems. Proceedings of the IEEE 89 (2), 174–186.

Minson, R., Theodoropoulos, G., 2004. Distributing RePast agent based simulations with HLA. In: Proceedings of the 2004 European Simulation Interoperability Workshop, Edinburgh.

Minson, R., Theodoropoulos, G., 2008. Distributing RePast agent-based simulations with HLA. Concurrency and Computation: Practice and Experience Journal. John Wiley & Sons. doi:10.1002/cpe.1280. 26p.

Morin, C., Puaut, I., 1997. A survey of recoverable distributed shared virtual memory systems. IEEE Transactions on Parallel and Distributed Systems 8 (9), 959–969.

Morse, K., Bic, L., Dillencourt, M., Tsai, K., 1999. Multicast grouping for dynamic data distribution management. In: Proceedings of the 31st Society for Computer Simulation Conference.

Nagaraj, S.V., 2004. Web Caching and its Applications. Springer, New York. 262p., ISBN: 1-4020-8049-2.

Ndiaye, S., Tsangou, M., Seck, M., Litwin, W., 2003. Range queries to scalable distributed data structure RP*. In: Proceedings of the Fifth Workshop on Distributed Data and Structures.

Oguara, T., Chen, D., Theodoropoulos, G.K., Logan, B., Lees, M., 2005. An adaptive load management mechanism for distributed simulation of multi-agent systems. In: Proceedings of the Ninth IEEE International Workshop on Distributed Simulation and Real-Time Applications, October 2005, pp. 179–186.

Pagel, B., Six, H.W., Toben, H., Widmayer, P., 1993. Towards an analysis of range query performance in spatial data structures. In: Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 214–221.

Perumalla, K.S., Fujimoto, R.M., Thakare, P.J., Pande, S., Karimabadi, H., Omelchenko, Y., Driscoll, J., 2005. Performance prediction of large-scale parallel discrete event models of physical systems. In: Proceedings of Winter Simulation Conference 2005.

Rajive, B., Deelman, E., Phan, T., 2001. Parallel simulation of large-scale parallel applications. The International Journal of High Performance Computing Applications 15 (1), 3–12.

Rak, S.J., Salisbury, M., MacDonald, R.S., 1997. HLA/RTI data distribution management in the synthetic theater of war. In: Proceedings of the Fall 1997 DIS Workshop on Simulation.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S., 2001. A scalable content-addressable network. Proceedings of the ACM SIGCOMM.

Riley, P., 2003. MPADES: Middleware for parallel agent discrete event simulation. In: Kaminka, G.A., Lima, P.U., Rojas, R. (Eds.), RoboCup-2002: The Fifth RoboCup Competitions and Conferences. Lecture Notes in Artificial Intelligence, vol. 2752. Springer, Berlin.

Riley, P., Riley, G., 2003. SPADES – a distributed agent simulation environment with software-in-the-loop execution. In: Chick, S., Sanchez, P.J., Ferrin, D., Morrice, D.J. (Eds.), Winter Simulation Conference Proceedings.

Samet, H., 1990. The Design and Analysis of Spatial Data Structures. Addison-Wesley, New York.

Steen, M.V., Hauck, F., Homburg, P., Tanenbaum, A.S., 1998. Locating objects in wide-area systems. IEEE Communications Magazine 36 (1), 104–109.

Stenström, P., 1990. A survey of cache coherence schemes for multiprocessors. ACM Computer 23 (6), 12–24. ISSN: 0018-9162.

Tanenbaum, A.S., 2003. Computer Networks, fourth ed. Prentice Hall, New Jersey, ISBN 0-13-066102-3.

Wang, F., Turner, S.J., Wang, L., 2003. Integrating agents into HLA-based distributed virtual environments. In: 4th Workshop on Agent-Based Simulation, Montpellier, France, pp. 9–14.