An Agent-based DDM for High Level Architecture*

Gary Tan and Liang Xu
School of Computing
National Univ of S'pore
Singapore 119260
gtan@comp.nus.edu.sg

Farshad Moradi
Swedish Defense Research Agency (FOI)
172 90 Sweden
farshad@foi.se

Simon Taylor
Dept of Info. Sys. & Computing
Brunel University
Uxbridge UB8 3PH, UK
simon.taylor@brunel.ac.uk

Abstract

The Data Distribution Management (DDM) service is one of the six services provided in the Runtime Infrastructure (RTI) of High Level Architecture (HLA). Its purpose is to perform data filtering and reduce irrelevant data communicated between federates. The two DDM schemes proposed for RTI, region-based and grid-based DDM, are oriented to send as little irrelevant data to subscribers as possible, but only manage to filter part of this information and some irrelevant data is still being communicated. In a previous paper [3], we employed intelligent agents to perform data filtering in HLA, implemented an agent-based DDM in RTI (ARTI) and compared it with the other two filtering mechanisms. This paper reports on additional experiments, results and analysis using two scenarios, the AWACS sensing aircraft simulation and the air traffic control simulation scenario. Experimental results show that compared with other mechanisms, the agent-based approach communicates only relevant data and minimizes network communication, and is also comparable in terms of time efficiency. Some guidelines on when the agent-based scheme can be used are also given.

1. Introduction

The High Level Architecture (HLA) provides the specification of a common technical architecture for modeling and simulation in the US Department of Defense (DoD), its primary goals being to facilitate interoperability among simulations and to promote re-use of simulations and their components. The HLA is composed of three major components: (i) HLA rules; (ii) HLA interface specification; and (iii) HLA object model template [1].

The services of the Runtime Infrastructure (RTI) are described by the HLA interface specification. The RTI is a collection of software that provides common services required by multiple simulation systems. These services fall into six categories: *federation management, object management, declaration management, ownership management , time management* and *data distribution management* [1].

The data distribution management (DDM) services are employed by federates to assert properties of their data or to specify their data requirements in terms of user-defined spaces. The DDM controls the efficient routing of information between federates to reduce the amount of irrelevant data sent between federates and cut network communication cost [1, 2]. The two traditional DDM mechanisms, i.e., the region-based mechanism and grid-based mechanism, are both oriented towards this purpose, but still suffer from several drawbacks. In a previous paper [3], we try to avoid these drawbacks by using agents to do accurate filtering so that the RTI routes only relevant information and minimizes communication. In our method, each time a federate subscribes to some data, intelligent mobile agents are launched to publishers of those data. These agents will fetch the updated data, perform data filtering and then send the subscriber the exact information that they require. We implemented an agent-based DDM in the RTI (ARTI) and compared it with the other two filtering mechanisms. This paper reports on additional experimental results and analysis using two scenarios, the AWACS sensing aircraft simulation scenario and the air traffic control simulation scenario.

This paper is organized as follows. Section 2 describes the two DDM mechanisms employed in RTI, while section 3 describes the agent-based DDM filtering mechanism and the design structure of ARTI. The implementation issues of ARTI are discussed in section 4. Section 5 describes the two experiment scenarios, the AWACS sensing aircraft and Air Traffic Control scenarios, used to compare the three DDM mechanisms. The experimental results and analyses are also provided in this section. The conclusion is given in section 6.

* This research is supported by the NUS-MINDEF collaboration GR6757

0-7695-1104-X/01 $10.00 © 2001 IEEE

2. DDM filtering mechanisms

The two main DDM data filtering mechanisms currently employed in RTI are the region-based filtering and the grid-based filtering, which are described briefly here. The reader is referred to [3] for a more detailed description.

2.1. Region-based filtering

The region-based filtering method uses a fundamental construct called the *routing space*. The routing space is a multi-dimensional coordinate system through which a region is specified. A federate tells the RTI to deliver only the data which fall within the extents of the region by specifying a subscription region. By specifying an update region with a particular object, a federate promises that the data within the extents of this region will be published when it tells the RTI to update its values/attributes.

When an update region and subscription region of different federates overlap, the RTI adds the subscribing federate into the receiving federate group connected to the update region. The data updated will be delivered to the subscribing federate. Each time an update region (or subscription region) is modified, the RTI does the matching again and accordingly updates the receiving federate group.

Every update region must be compared with all the subscription regions to determine if they should be updated. If the number of subscription regions and update regions is high, the number of matches is also high.

2.2. Grid-based filtering

In grid-based filtering, the routing space is partitioned into a grid of cells. Cells are used to efficiently specify the overlapped part between a subscription region and an update region. For each cell, a multicast group containing subscribers whose regions overlap with that cell is maintained. When a publisher's update region overlaps with a cell, the data associated with the update region will then be delivered to the multicast group for that cell. As in the region-based approach, during the simulation runtime execution, the RTI will perform matching again and adjust the multicast groups of the cells when it is notified that a region is modified.

One problem with this method is that irrelevant data may be sent due to subscription and update regions of federate overlapping with a cell, but do not themselves overlap.

2.3. Drawbacks of current approaches

Before DDM was introduced in the RTI, the data distribution in HLA was based mainly on *class* specification by employing the Declaration Management (DM) services. Irrelevant information was generally discarded at the receiving simulations in order to reduce local processing loads. The region-based filtering mechanism and grid-based filtering mechanisms define federates' interest in data as value-based. They however have some obvious drawbacks. These are excessive cost of critical network communication resources because of irrelevant information, and inexact routing causing extra processing in filtering at the receiving simulations' side.

2.4. Related work

There has been much research done on the two main DDM mechanisms, region-based DDM mechanism and grid-based DDM mechanism. For example, Cohen and Kemkes discussed various ways of using the DDM services and studied their impact on performance [4]. Rak and Van Hook gave evaluation of the grid-based relevance filtering [5]. Rizik's research work [6] tried to find an optimal geographic routing space cell size for a specific simulation model. Boukerche et al [7] uses a fixed grid, sender-based hybrid to reduce the multicast groups to improve performance.

In a previous paper [3], a new DDM filtering mechanism based on agents was proposed. This mechanism attempts to overcome the drawbacks discussed previously. Influenced by the popularity of intelligent agents, we noticed that the various properties of agents, including autonomy, mobility, intelligence, sociability and etc, may help to improve the efficiency of the data filtering in the DDM and to overcome these drawbacks. We incorporated the agent technology to the DDM approach and implemented an agent-based DDM filtering mechanism.

In this agent-based filtering mechanism, the RTI delivers to data consumers exactly what they want by using agents to perform exact data filtering in HLA. No irrelevant information will be distributed in this new DDM mechanism. Thus, the network communication is minimized and the extra cost of the local processing resources consumed to filter out irrelevant data is eliminated.

Employing agents in the RTI data distribution framework is previously discussed by Van Hook et al [8, 9]. But in their work, agents are mostly used as an auxiliary part of the data distribution system to collect some useful information and to help form the multicast group. One of their papers [9] mentions that mobile agents may go to publishers' LRC from subscribers' to fetch relevant data, but it gives no detail or method.

The simulation group at Brunel uses a novel agent-based service distribution model called *Thin Agents* to reduce network bandwidth in GRIDS, a generic run-time infrastructure for distributed simulation [10].

3. Agent-based DDM

3.1. Architecture

The description of the agent-based RTI (ARTI) can be found in [3]. We briefly describe the architecture here again for clarity and completeness. We employed a totally distributed RTI structure and the RTI services are implemented in the RTI ambassador of each federate. The design structure of the ARTI is shown in figure 1.

In the HLA, federates make subscriptions to declare their interest in the object attributes or interactions that other federates own. When a subscription is made, intelligent mobile agents are launched to the publishers of those attributes and interactions. Whenever the publishers update their object attributes or interactions, those agents associated with them will fetch the data, perform data filtering and then send the subscriber exactly what it wants. At the same time, agents keep their internal filtering parameters updated by receiving notification from the subscribing federate once a region modification is made.

When a federate calls the DDM services to subscribe to some object class attributes or interaction classes with a region, *Sub_child*ren are launched to the places where they are most likely to find those data. That means the *Sub_child*ren are sent out to the federates which publish those data, according to the internal record of the system's publication information. A *SubMaster_agent* is built to manage those *Sub_child*ren.

When they have reached the (publishing) federates, *Sub_child*ren communicate directly with the federate environment, fetch the published/updated data, perform filtering for their owners (i.e., the subscribers which launch them), then send the filtered results to the subscribers.

In the ARTI, federates can change their subscription regions by notifying their *Sub_child*ren of the new subscription region. Upon receiving the notification of region modification, the *Sub_child*ren modify their internal filtering parameters accordingly. *Sub_child*ren employ the new filtering parameters the next time they perform data filtering. As a result, only the information that falls into the new subscription region will be routed to the subscribing federate.

It should be noted that no update regions exist in the agent-based DDM. Since the agent deals with all the updated data and deliver the data to its owner federate if the data exactly fall into the subscription region of its owner federates, no update region needs to be defined to clarify the area where updated data will fall into. It is different from the situation in region-based DDM and grid-based DDM, where the data receiving federate group is decided by the matching result between update and subscription regions.

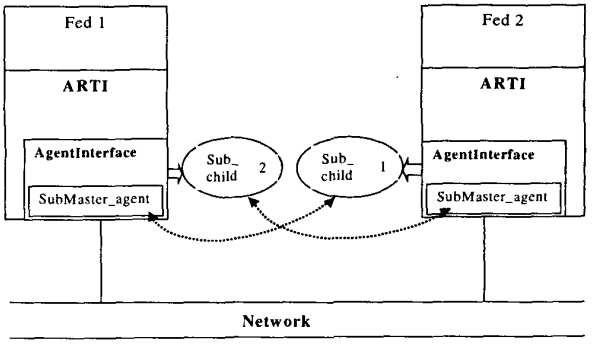

Figure 1: Design structure of agent-based RTI

3.2. Discussion

Compared with the other data filtering mechanisms, by incorporating agent technology in the RTI,

- Extra processing which has been devoted to filtering by the receivers now is done by agents at the physical location of senders, i.e., the publishing federates, on behalf of the receivers.
- Exact filtering by agents at the location of senders leads to exact routing for the data from publisher to subscriber.
- Critical network communication is minimized, by communicating only relevant data.

However, the agent-based DDM mechanism has some disadvantages. Firstly, the internal table of other federates' publishing interest costs storage space. But compared with the changing update regions and subscription regions, it is only a very small cost. Secondly, launching agents to other federates needs a period of time which may delay the program execution. Fortunately, agents are launched only in the initialization period of a simulation, and it will not affect the federation execution progress.

Another disadvantage of the agent-based DDM occurs in large-scale simulations. For a federation with large numbers of federates, hundreds or thousands of filtering agents may be sent out to one federate as a result of the initial matching operation based on the DM services. These large numbers of filtering agents, residing on one federate machine and performing filtering from time to time, will cause a heavy load to the federate machine. The agent-based DDM mechanism will become very inefficient in such cases of large-scale simulations.

4. Implementation issues

Our implementation is built on an MPP – the Fujitsu AP3000/32, UNIX (r) System V, Release 4.0, and the language used is C/C++. The communication

architecture is TCP based on the Fast Messages (FM) interface provided by the Federated Simulations Development Kit (FDK RTI-Kit) [11]. The agents are created and act in the environment of D'Agents 2.0 [12]. The Federated Simulations Development Kit (FDK) is a software system that has been developed at Georgia Tech. It comprises composable modules for building RTIs. D'Agents is a mobile-agent system that is under development at Dartmouth College and is used in several information-retrieval and work-flow applications. D'Agents is based on the scripting language Tcl7.5 and Tk4.1 and runs on standard Unix platforms.

The D'Agents system is still under development. Currently, the agents' communication is restricted to between agents, and agents cannot communicate freely with other environment resources. The primitives for agents' activity are also limited. This causes our agents to be not as "intelligent" as it is desired especially when they act against the changes of surrounding environment and communicate with it. Because of this limitation, we employ the socket method for the communication between agents and other processing environment. A federate process writes data to a socket, and the agents fetch data from the socket.

Another implementation technical problem is that the only variable format supported by Tcl agents (D'Agents) is STRING, as TCL/TK is a descriptive language. We have to transform all the data of complex C/C++ data structures to strings before transferring them to Tcl agents, and vice versa. This method is slow and inefficient and will degrade performance.

Because of the above technical problems, currently the ARTI still has some limitations, especially in the time latency when reflecting updated information at the subscriber federates. These limitations need to be overcome in the future by either improving the agent interface, or by writing a custom-built agent tool-kit.

5. Experiment Scenario

The AWACS sensing aircraft [4] simulation scenario and Air Traffic Control (ATC) [14] simulation scenario were used to test the ARTI and to compare the efficiencies of agent-based DDM filtering mechanism with the other two traditional DDM mechanisms.

5.1. Experiment scenario 1: AWACS

In the AWACS sensing aircraft scenario, as described in [4], there are several aircraft and an AWACS circling over an area. When some aircraft happens to fly into the sensor range of the AWACS, it is discovered by the AWACS and its position is noticed by the AWACS.

5.1.1. Experiment assumptions

We assume all aircraft and the AWACS fly at different randomly generated speeds from a randomly generated initial position. Two federates are used, one represents the AWACS, and the other represents the aircraft. The subscription region of the AWACS can be defined from its sensor range. In region-based and grid-based DDM mechanisms, an update region is defined around each aircraft's position (there is no need for an update region in the agent-based filtering mechanism, as agents do exact filtering at the physical locations of the publishers). The AWACS federate subscribes to the aircraft information, but does not publish anything, and the aircraft federate publishes its position, but does not subscribe to any data.

The scenario is performed as a time-stepped simulation [13]. What the simulator calculates is the system status at each time step but not the continuous system moving process. Therefore the update and subscription regions have to be carefully defined to avoid missing the objects falling into the AWACS sensor range in the period between two discrete time steps. In our geographic routing space, we define the subscription region to be equal to the sensor region plus the maximum moving distance of the AWACS during the time between two subsequent subscription region modification requests. The update region is equal to the aircraft's maximum range during the time between two subsequent update region modification requests.

In the simulation, the object attribute values (i.e., the positions of the aircraft) are updated at each time step, but the subscription region and the update regions (only for region-based and grid-based DDM mechanisms) are modified once per k time steps. The frequency of region modification depends on the value of k.

5.1.2. Experimental results

Simulations are performed using ARTI, RTI with only DM, RTI with region-based DDM and RTI with grid-based DDM using different k values and the number of communicated messages after filtering is observed. The simulations are run on a Fujitsu AP3000 32-node multiprocessor running Unix (r) System V, Release 4.0. The AWACS federate and aircraft federate are run on different nodes. Both federates are written in C++.

Table 1 gives the parameter sets used in the simulations (for set A: k =1, set B: k=2, and set C: k=3). The speeds of AWACS and aircraft are generated randomly based on a fixed maximum speed. Their initial positions are also randomly generated in the initialization procedure. When testing the grid-based filtering mechanism, the routing space is divided into a number of grid-cells.

Parameter	Value		
No. of Aircraft	40		
Max Speed - Aircraft	900.0 (km/h)		
Max Speed -AWACS	450.0 (km/h)		
Sensor Range	15.0 (km)		
Subscription Region (km^2)	2025.0 (k=1)	8100.0 (k=2)	22500.0 (k=3)
Routing Space	600*600(km^2)		
Simulation Time	100 (time-steps)		

Table 1: Parameter table

5.1.3. DM vs. agent-based DDM

A simple comparison is made between the agent-based DDM and the DM, which employs only a class-based filtering without any DDM mechanisms. Table 2 gives the test results. The simulation period is 100 time steps. K is set to 3 in the agent-based filtering mechanism. Without employing any DDM filtering mechanism, the DM is no doubt much less efficient than the agent-based DDM and communicates much more redundant messages.

Filtering mechanism	No. of messages
DM	4000
Agent-based DDM (k=3)	146

Table 2: Results of DM vs. agent-based DDM

5.1.4. Grid vs. region vs. agent-based

Simulations were conducted using the three DDM mechanisms at different k values. For the grid-based DDM, since its filtering efficiency largely depends on the cell size, we tested different cell sizes and chose the one that gave the least number of communicated messages [3]. Because at each run of the simulation the initial positions and speeds of all the objects are randomly generated, the filtering results for each run of simulation are not the same. We therefore took replications for each parameter set and calculated the average filtering results. Figure 2 shows the results of the three DDM mechanisms.

5.1.5. Analysis

As seen in the figure, as k increases, the number of communicated messages of both the region-based and grid-based DDM increases. This is because the subscription region and the update region become larger when k increases, thus increasing the possibility of area overlap. This results in more messages being sent.

From the figure we can clearly see that much less information is sent when using the agent-based DDM filtering mechanism than by using grid-based or region-based DDM mechanisms. The grid-based DDM and region-based DDM may deliver irrelevant information in addition to the data the subscribers require. However, the agent-based DDM communicates only relevant data and not redundant information. Thus, the agent-based DDM mechanism improves the filtering efficiency of DDM and minimizes network communication.

The number of communicated messages of agent-based DDM for the different k values as shown in figure 2 are similar and increases slowly when k increases from 1 to 3. As in the cases of the region-based and grid-based DDM, the increasing messages result from the increasing subscription region. As the subscription region becomes larger, the possibility for the aircraft falling into the subscription region increases. However, since the agent-based DDM routes only the required messages, the numbers of routed messages do not vary much for different k values for the same simulation scenario.

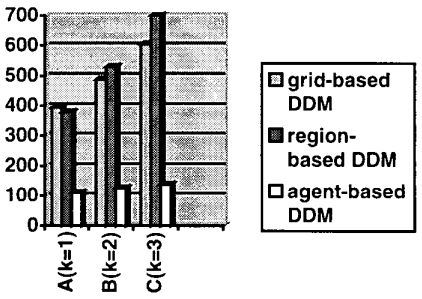

Figure 2: Number of communicated messages for parameter sets A, B, C

The following table 3 shows the simulation processing times of running the simulation with different DDM mechanisms for k = 1. These processing times are collected by running the simulations exclusively on the nodes of Fujitsu AP3000, using the unix command *time*. Compared with the other two mechanisms, the simulation with agent-based DDM takes a longer time. This time latency comes mainly from the time costs for launching agents in the initial period of simulation and the technical problems we mentioned earlier in section 4. Although the simulation with agent-based DDM has a longer processing time than both the grid-based and region-based DDM, they are at least of the same time order. We anticipate that an improvement can be expected if we can address these problems. From table 3, we see that the grid-based DDM mechanism takes more time than the region-based DDM mechanism. This can be attributable to the processing cost of updating the multicast groups of the cells when both the update regions and subscription regions change dynamically over time in the grid-based DDM.

DDM mechanism	Simulation processing time (s)
Agent-based	30.513
Grid-based	27.914
Region-based	23.537

Table 3: Simulation processing times (set A)

5.2. Experiment scenario 2: air traffic control

The air traffic control simulator is a non-military HLA federation designed and developed to simulate air traffic control (ATC) in a distributed environment [14]. In this simulation model, each airport (denoted by APi in figure 3) is represented by a federate and controls a set of aircraft objects. A radar is located at each airport and the radar range (denoted by Ri) covers a portion of the routing space. The aircraft (denoted by Ai) stays at an airport for a random period of time, takes off, flies and lands at its destination airport. At the destination, it stays again for a random period of time, selects the next destination, takes off and lands. This process is performed repeatedly. When some aircraft flies into the radar range of any airport other than its own controller, it is discovered and its position is noted by the airport.

This scenario differs from the first one in that the subscribers (airports) are static and hence their subscription regions do not change.

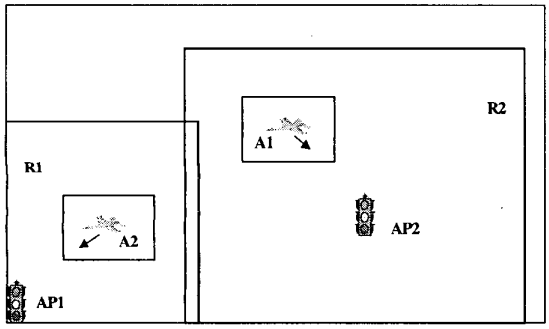

Figure 3: Air traffic control scenario

5.2.1. Experiment assumptions

In the experiment, we assume a simulation scenario of two airports (AP1 and AP2), where each airport controls one aircraft. Two federates are designed for this scenario, each federate representing one airport. The objects (the aircraft) fly at a fixed speed, 100 km per time step, starting from their controller airport. The radar range of the airport is defined by the subscription region of the airport federate. In the region-based and grid-based DDM mechanisms, the update region is defined around the aircraft's current position. The airport federate subscribes to the information of the aircraft controlled by the other federate with its radar range. At the same time, it publishes the position of the aircraft that it controls.

For the example depicted in figure 3, AP1 has control of aircraft A1 and AP2 has control of A2. AP1 subscribes to A2's position with its subscription region R1. At the same time, AP1 publishes the position of aircraft A1. Also, AP2 subscribes to A1's position with its subscription region R2, and at the same time, it publishes the position of aircraft A2.

The experiment is performed as a time-stepped simulation [13]. In the routing space, we define the update region equal to the aircraft's maximum moving range during the time between two subsequent update region modification requests. The subscription region is the radar range of the airport.

In the simulation, the object attribute values (i.e., the positions of the aircraft) are updated at each time step, but the update regions (only for region-based and grid-based DDM mechanisms) are modified once per k time steps. The frequency of update region modification depends on the value of k. The subscription region is unchanged during the simulation.

During the air traffic control simulation, the radar ranges of the airports are unchanged. Thus after the initial setting up of the subscription region, there is no need to modify the subscription region of the airport during the simulation loop. This is different from the algorithm for the AWACS sensing aircraft scenario, where the subscription region of the moving AWACS is changed once for each k time steps.

5.2.2. Experimental results

The same set of experiments is performed using this scenario. The airport federates are run on different nodes, the aircraft are considered objects owned by the airport federates.

Table 4 gives the parameters we used in the simulations. The initial positions of the aircraft are set in the initialization procedure as the positions of their controller airport. When testing the grid-based filtering mechanism, the routing space is divided into a number of grid-cells. Different values of k are used in the tests for region-based and grid-based DDM mechanisms.

Routing Space (km^2)	10000.0*10000.0
Positions of airports (km, km, km)	AP1(10.0, 0.0, 0.0); AP2(7500.0, 2000.0, 0.0)
Radar Range (km)	4000.0
Subscription region(km^2)	4000.0*4000.0
Speed of Aircraft (km/time step)	100.0
Sim Time (time steps)	300
k (time steps)	k=1, 2, 3

Table 4: Parameter table for ATC

5.2.3. DM vs. Agent-based DDM

The agent-based DDM is first compared with the DM. The results when run for different simulation times are listed in table 5. From the table it is easy to see that the agent-based DDM again communicates much less messages than the DM (about 50%). Agents in the agent-based DDM send an airplane's attributes only when the airplane is in the radar range of the subscribing airport. This is roughly half the time of the simulation, since the aricraft fly from one airport to the other airport.

Simulation time (Time steps)	300		200		100	
DM	300	299	200	199	100	99
Agent-based (ADDM)	168	161	96	99	58	57
ADDM/DM(%)	56	54	48	50	58	58

messages received by AP1 messages received by AP2

Table 5: Results of DM vs. agent-based DDM

5.2.4. Grid vs. region vs. agent-based

We performed the air traffic control simulation using the three DDM mechanisms using different k values. As in the case of the AWACS sensing aircraft scenario, for grid-based DDM, we divide the routing space into different numbers of cells, and choose the one that gives the least number of communicated messages.

5.2.5. Analysis

Figure 4 illustrates and summarizes the number of received messages by AP1 and AP2 after filtering using the grid-based, region-based and agent-based DDM filtering mechanisms. For either airport, less information is delivered by using the agent-based DDM filtering mechanism than by using the grid-based or the region-based DDM mechanisms. It is again shown that the agent-based DDM mechanism uses minimal communication network resources by sending only relevant information.

In the air traffic control scenario, although the number of communication messages in the agent-based DDM is less than in the grid-based DDM and the region-based DDM, the improvement is not that great. This is because of the unchanged subscription region and the aircraft's line of flight. The subscription regions are kept unchanged during the simulation time. Only when the aircraft flies close to the subscription region of its subscribing airport will its update region overlap with the subscription region, and the data of its attributes are then delivered. Soon after this, the aircraft enters the radar region, agents in agent-based DDM route the relevant data of its attributes. Thus the time when the RTI begins to deliver the required information in the grid-based DDM and the region-based DDM is close to the time in the agent-based DDM. So is the time when RTI stops delivering information when the aircraft leaves a subscription region. In such cases, the three mechanisms differ little in the number of communicated messages. However, in the grid-based and region-based mechanisms, the frequent modification of update regions is still a big cost, but does not give rise to any additional network messages.

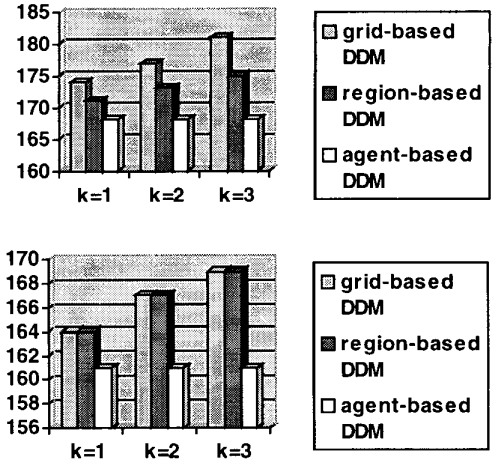

Figure 4: Received messages by AP1 (top) & AP2 (bottom) using different k values

Table 6 gives the simulation processing times of running the air traffic control simulation with the different DDM mechanisms for k = 1. These times are collected by running the simulations exclusively on the nodes of Fujitsu AP3000, using the unix command *time*. The timing trend is similar to the AWACS sensing aircraft scenario. The simulation using the agent-based DDM runs for a longer time than the grid-based and the region-based DDM mechanisms. Although it performs worse than the grid-based DDM and takes twice as long as the region-based one, it is still of the same time order. As we mentioned in the AWACS sensing aircraft scenario, this time latency is attributable to the inefficiencies of interfacing the agents with the RTI environment. Improvements for the agent-based mechanism can be expected by improving this interface.

Another reason is because in this scenario, the subscription regions do not change, and they are much bigger than the update regions. The update regions also overlap with the subscription regions for approximately half the time, since the aircraft fly between airports. For the grid-based DDM, this translates to less frequent updates of the multicast groups, and hence less processing overheads. For the region-based case, the number of matching is low due to the small number of objects.

DDM mechanism	Simulation processing time (s)
Agent-based	15.992
Grid-based	11.612
Region-based	7.028

Table 6: Simulation processing times for k = 1

6. Conclusion

The agent-based DDM filtering mechanism has been proposed to overcome the drawbacks of region-based and grid-based DDM by filtering only relevant information. This paper has compared the three DDM filtering schemes using two different scenarios, the AWACS sensing aircraft and the air traffic control simulation scenarios. In the AWACS sensing aircraft scenario, the experimental results showed that the communicated messages using the agent-based DDM were significantly less than the messages of the region-based and grid-based DDMs (less than one third of the communicated messages of the other two mechanisms). The agent-based DDM was much more efficient in filtering, routing only relevant information and minimizing the network communication. In the air traffic control simulation scenario, where the subscription regions were large and remain fixed, and the number of objects was small, the benefit of the agent-based DDM was not that great compared to the first scenario.

The agent-based DDM mechanism spends more processing time than the other two mechanisms, attributable to the inefficiency of the agent interface. An improvement in time cost can be expected by improving this interface. The performance is worse in the second scenario where the overlap frequency between subscription and update regions was low.

Thus the agent-based DDM is useful for scenarios where the subscription regions change dynamically over time, and the update and subscription region overlap frequency is high. In such cases, the overheads associated with updating multicast lists and sending irrelevant messages will come into play.

During the implementation and testing of ARTI, we observed that the information reflection at the subscriber federates is delayed. This is mostly caused by the technical problems discussed in section 4. The time latency for information reflection limits the application of the agent-based DDM in real time systems. In addition, the agents in our implementation model can be more "intelligent" and more flexible, providing a more complete agent environment tool.

Other further work concerns the scalability of the agent-based DDM. For the agent-based scheme to be viable, its scalability when used in large-scale simulations must be investigated. Improvements need to be made on the design of ARTI to answer this question.

7. References

1. U.S. Department of Defense, "High Level Architecture Interface Specification", Version 1.3, 2 April 1998. http://www.dmso.mil/
2. Katherine L. Morse, Jeffrey S. Steinman, "Data Distribution Management in the HLA, Multidimensional Regions and Physically Correct Filtering", in Proceedings of the Simulation Interoperability Workshop, Spring 1997.
3. G.Tan, L.Xu, F.Moradi and YS Zhang, "An Agent-based DDM Filtering Mechanism", in proceedings of MASCOTS, San Francisco, USA, Aug 2000.
4. Danny Cohen and Andreas Kemkes, "User-Level Measurement of DDM Scenarios", in proceeding of the Simulation Interoperability Workshop (SIW), Spring 1997.
5. Steven J. Rak and Daniel J. Van Hook, "Evaluation of grid-based relevance filtering for multicast group assignment", in Proceedings of the Distributed Interactive Simulation, 1996.
6. Pete Rizik et al., "Optimal geographic routing space cell size in the FEDEP for prey-centric models", in Proceedings of the Simulation Interoperability Workshop (SIW), Spring 1998.
7. A.Boukerche, A.Roy & N.Thomas, "Dynamic Grid-Based Multicast Group Assignment in Data Distribution Management", proceedings of 4th Workshop on Distributed Simulation and Real-time Applications, San Francisco, USA, August 2000.
8. Daniel J. Van Hook, Steven J. Rak, James O. Calvin, "Approaches to RTI Implementation of HLA Data Distribution Management Services", the 15th Workshop on Standards for the Interoperability of Distributed Simulations, September 1996.
9. Van Hook, Daniel J., David P. Cebula, Steven J. Rak, Carol J. Chiang, Paul N. DiCaprio, James O. Calvin, "Performance of STOW RITN Application Control Techniques", 14th Workshop on Standards for the Interoperability of Distributed Simulations, March 1996.
10. R.Sudra, T.Janahan, S.Taylor, "Distributed Supply Chain Simulation in GRIDS", proceedings of 2000 Winter Simulation Conference, Florida, U.S.A., December 2000.
11. Georgia Tech Research Corporation, "FDK-Federated Simulations Development Kit", November 1998. http://www.cc.gatech.edu/computing/pads/fdk.html
12. Dartmouth College, "D'Agents", November 1999. http://agent.cs.dartmouth.edu
13. Gary Tan, Rassul Ayani, Yusong Zhang and Farshad Moradi. "Grid-based Data Management in Distributed Simulation", in Proceedings of 33rd Annual Simulation Symposium, Washington, U.S.A., April 2000.
14. Farshad Moradi, Rassul Ayani, Gary Tan, "Object and Ownership Management in Air Traffic Control Simulations", in Proceedings of IEEE Distributed Interactive Simulation –Real Time '99, Maryland, U.S.A., October 1999.