DYNAMIC INTEREST MANAGEMENT IN THE DISTRIBUTED SIMULATION OF AGENT-BASED SYSTEMS

Brian Logan
School of Computer Science and IT
University of Nottingham
Nottingham NG8 1BB, UK
bsl@cs.nott.ac.uk

Georgios Theodoropoulos
School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK
gkt@cs.bham.ac.uk

KEYWORDS:agents, simulation, interest management, load balancing.

ABSTRACT

Agent-based systems are increasingly being applied in a wide range of areas including telecommunications, business process modelling, computer games, control of mobile robots and military simulations. Such systems are typically extremely complex and it is often useful to be able to simulate an agent-based system to learn more about its behaviour or investigate the implications of alternative architectures. However conventional distributed simulation techniques cannot easily be applied to the problem of modelling the interaction of a system of autonomous components. In this paper, we identify the efficient distribution of the agents' environment as a key problem in the parallel distributed simulation of agent-based systems and propose an approach to the decomposition of the environment which facilitates load balancing.

INTRODUCTION

Agent-based systems are increasingly being applied in a wide range of areas including telecommunications, business process modelling, computer games, control of mobile robots and military simulations (Bradshaw 1997, Jennings & Wooldridge 1998). Such systems are often extremely complex and it can be difficult to formally verify their properties. It is therefore useful to be able to *simulate* an agent-based system to learn more about its behaviour or investigate the implications of alternative architectures.

However, the computational requirements of simulations of agent-based systems far exceed the capabilities of conventional sequential von Neumann computer systems. Each agent is typically a complex system in its own right (e.g., with sensing, planning, inference etc. capabilities), requiring considerable computational resources, and many agents may be required to investigate the behaviour of the system as a whole or even the behaviour of a single agent (Sloman 1998). One solution to this problem is to attempt to exploit the high degree of parallelism inherent in agent-based systems. However the application of conventional distributed simulation techniques to the problem of modelling the interaction of a system of autonomous components is not straightforward.

In this paper, we identify the efficient distribution of the agents' environment as a key problem in the parallel distributed simulation of agent-based systems and propose an approach to the decomposition of the environment which facilitates load balancing.

MODELLING MULTI-AGENT SYSTEMS

The term *agent* is used in many different senses in the agent-based systems literature (see, for example, (Franklin & Graesser 1997)). For the purposes of this paper, we shall define an *agent* as a self-contained, concurrently executing thread of control that encapsulates some state and communicates with its environment and possibly other agents via some sort of message passing (Wooldridge & Jennings 1995).

There are many approaches to constructing agent-based systems, and many different architectures have been proposed. In many cases, these architectures offer considerable opportunities for parallel simulation in their own right. However, for ease of exposition, we will assume that an agent can be modelled as a single *Agent Logical Process* (ALP), and will not consider

simulation of the agent's internal operation further.

Agents are embedded in an environment. The *environment* of an agent is that part of the world or computational system 'inhabited' by the agent. The environment may contain other agents whose environments are disjoint with or only partially overlap with the environment of a given agent. We assume that objects and processes within the agents' environment are modelled as one or more *Environment Logical Processes* (ELP). There are many ways in which the agents' environment can be decomposed into ELPs. The appropriate 'grain size' of the simulation will depend both on the application and on practical considerations, such as the availability of existing simulation code. While there are obvious advantages in reusing part or all of an existing simulation, modelling the environment as a single logical process can create a bottleneck in the simulation which degrades its performance, since all all agents must react to and act within the environment.

Different kinds of agent have differing degrees of access to different parts of the environment. For example, a WWW agent has in principle complete access to any site on the INTERNET. Conversely, a synthetic pilot agent in a military training simulation typically only has access to a small part of its environment. The degree of access is dependent on the type and range of the agent's sensors and the actions it can perform. Moreover, in many cases, an agent's actions can effectively change the topology of the environment.

It is therefore difficult to determine an appropriate simulation topology *a priori*. As a result, a simulation of a multi-agent system typically requires a (very) large set of shared variables which could, in principle, be accessed or updated by the agents (if they were in the right position at the right time etc.). Which variables the agents can in fact access/update depends both on the state of the agents (e.g., their position) and the state of the rest of the environment. However, the information required to determine the set of variables which will be affected by an event is itself distributed. The potentially all-to-all communication of the shared state variables results in broadcast communication, which is extremely costly and limits any gains which may be achieved through distributed simulation.

INTEREST MANAGEMENT

The problem of avoiding broadcast communication has been addressed mainly in the context of real time large scale simulations where it is termed Interest Management (Morse 1996). Interest Management techniques utilise filtering mechanisms based on *interest expressions (IEs)* to provide the processes in the simulation with only that subset of information which is relevant to them (e.g., based on their location or other application-specific attributes).

Various Interest Management schemes have been devised, utilising different communication models and filtering schemes (Morse 1996). In most existing systems, Interest Management is realised via the use of IP multicast addressing, whereby data is sent to a selected subnet of all potential receivers. A multicast group is defined for each message (PDU) type, grid cell (spatial location) or region in a multidimensional parameter space in the simulation. Typically, the definition of the multicast groups of receivers is static, based on a priori knowledge of communication patterns between the processes in the Simulation (Smith, Russo & Schuette 1995, Mastaglio & Callahan 1995, Macedonia, Zyda, Pratt & Barham 1995, Calvin, Chiang & Van Hook 1995, Steinman & Weiland 1994). The HLA[1] utilises the *routing space* construct, a multi-dimensional coordinate system whereby simulation federates express their interest in receiving data (subscription regions) or declare their responsibility publishing data (update regions). In existing implementations, the routing space is subdivided into a predefined array of fixed size cells and each grid cell is assigned a multicast group which remains static and fixed throughout the simulation; a process joins those multicast groups whose associated grid cells overlap the process subscription region.

Static, grid-based Interest Management schemes have the disadvantage that they do not adapt to the dynamic changes in the communication patterns between the processes during the simulation. Furthermore, in order to filter out all irrelevant data, grid-based filtering requires a reduced cell size, which in turn implies an increase in the number of multicast groups, a limited resource with high management overhead. More recently, there have been a few attempts to define alternative dynamic schemes for Interest Management concentrating mainly on the dynamic configuration of multicast groups within the context of HLA. For instance, Berrached et al. (Berrached, Beheshti, Sirisaengtaksin & deKorvin 1998) examine hierarchical grid implementations and a hybrid grid/clustering

[1] In the context of the High Level Architecture, Interest Management is also referred to as Data Distribution Management (DDM)

scheme of update regions to dynamically reconfigure multicast groups while Morse et al. (Morse, Bic, Dillencourt & Tsai 1999) report on preliminary investigations on a dynamic algorithm for dynamic multicast grouping for HLA. Saville et al. (Saville 1997) describe GRIDS, a generic runtime infrastructure which utilises dynamic instantiation of Java classes in order to achieve interest management.

In this paper we present a new approach to dynamic Interest Management, which targets the simulation of agent-based systems. Our approach is not confined to grids and rectangular regions of multidimensional parameter space and does not rely on the support provided by the TCP/IP protocols. It is based on the notion of *spheres of influence* (Theodoropoulos & Logan 1999), which are used to dynamically decompose and distribute the shared state so that bottlenecks and broadcast communication are minimised. Furthermore, our approach aims to exploit this decomposition in order to perform load balancing in dynamic, distributed architectures, an issue which to date has received little attention in the context of large scale simulation systems which execute on fairly static LAN/WAN configurations.

SPHERES OF INFLUENCE

We assume that each ALP/ELP is capable of generating and responding to at most a finite number of event types. Different types of events will typically have different effects on the agents and the environment, and, in general, events of a given type will affect only certain types of state variables (all other things being equal).

We define the *sphere of influence* of an event as the set of state variables read or updated as a consequence of the event. The sphere of influence depends on the type of event (e.g., sensor events or motion events), the state of the agent which generated the event (e.g., its position in space in the case of a robot, or the machines to which it currently has a network connection in the case of a WWW agent) and the state of the environment. The sphere of influence of an event is limited to the *immediate* and *predictable* consequences of the event rather than its ultimate effects, which depend both on the current configuration of the environment and the actions of other agents in response to the event.

We can use the spheres of influence of the events generated by each agent to derive an idealised decomposition of the shared state into logical processes. We define the sphere of influence of an agent a_j over the time interval $[t_1, t_2]$, $s(a_j, [t_1, t_2])$, as the union of the spheres of influence of the events generated by the agent over the interval. Intersecting the spheres of influence for each event generated by the agent gives a partial order over sets of state variables for the agent over the interval $[t_1, t_2]$, in which those sets of variables which have been accessed by the largest number of events come first, followed by those less frequently accessed, and so on. The rank of a variable v_i for agent a_j over the interval $[t_1, t_2]$, $r(v_i, a_j, [t_1, t_2])$ is the number of events in whose sphere of influence v_i lies.

Intersecting the spheres of influence for each agent gives a partial order over sets of state variables, the least elements of which are those sets of state variables which have been accessed by the largest groups of agents. This partial order can be seen as a measure of the difficulty of associating variables with a particular agent: the state variables which are members of the sets which are first in the order are required by the largest number of agents, whereas those sets of state variables which come last are required by only a single agent.

Any approach to the decomposition of the shared state into logical processes should, insofar as is possible, reflect this ordering. However, any implementation can only approximate this idealised decomposition, since calculating it requires information about the global environment, and obtaining this information would not be efficient in a distributed environment. Moreover, this ordering will change with time, as the state of the environment and the relative number of events of each type produced by the agents changes, and any implementation will have to trade off the cost of reorganising the tree to reflect the ideal decomposition against the increase in communication costs due to increased broadcast communication.

DISTRIBUTING THE STATE

The decomposition of the state is achieved by means of an additional set of Logical Processes, namely *Communication Logical Processes* (*CLPs*). The CLPs act as Interest Managers. Each CLP maintains a subset of the state variables and the interaction of ALPs and ELPs is via the variables maintained by the CLPs. CLPs enable the clustering of ALPs and ELPs with overlapping spheres of influence and facilitate load balancing. The partitioning of the shared state is performed dynamically, in response to the events generated by the ALPs

and ELPs in the simulation. Thus, the number and distribution of CLPs is not fixed, but varies during the simulation.

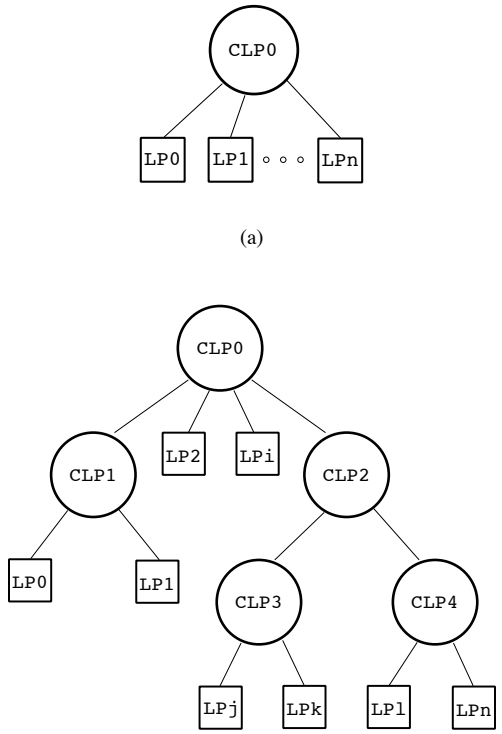

Figure 1: Generating the tree of CLPs.

We now sketch an algorithm for the decomposition of the shared state into CLPs. Initially, the whole of the shared state is handled by a single CLP, as depicted in Figure 1(a). All read and update events from all ALPs and ELPs are all directed to this single CLP, as is all inter-agent communication.

As simulation progresses, the CLP performs a dynamic analysis of the pattern and frequency of state accesses and computes an approximation of the agents' spheres of influence. If the load increases to the point that the CLP becomes a bottleneck (e.g., when message traffic exceeds a predefined threshold), the CLP creates one or more new CLPs, to which it assigns those disjoint subsets of the state variables that form the least elements in its approximation to the partial order over the spheres of influence. Those groups of ALPs and ELPs whose events and actions have formulated the new CLP(s) communicate directly with the corresponding new CLP. The process then repeats with the newly created CLP(s) monitoring the load and generating additional CLPs as required to keep the overall simulation load on the CLPs within bounds (Figure 1(b)).

This behaviour naturally leads to a tree structure, where the ALPs/ELPs are the leaves and the CLPs the intermediate nodes of the tree. Events by the ALPs/ELPs which refer to state variables not maintained by their parent CLP will be routed through the tree to the appropriate CLP node. This can be accomplished by recording in each CLP routing information specifying which event types are relevant to its child ELPs, ALPs and CLPs and to its parent CLP.

We define the cost of accessing a variable v_i for an agent a_j as the rank of v_i for a_j, $r(v_i, a_j, [t_1, t_2])$, times the number of CLPs which must be traversed to reach v_i during the interval $[t_1, t_2]$, $l(v_i, a_j, [t_1, t_2])$, i.e., the cost of accessing variables in the local CLP is 0. Then the cost to an agent a_j of accessing all the variables in the agent's sphere of influence $s(a_j, [t_1, t_2])$ is:

$$\sum_{v_i \in s(a_j, [t_1, t_2])} l(v_i, a_j, [t_1, t_2]) \times r(v_i, a_j, [t_1, t_2])$$

and the total access cost for all agents $a_1, \ldots, a_n$ of a particular decomposition over the interval $[t_1, t_2]$ is:

$$\sum_{j=1}^{n} \sum_{v_i \in s(a_j, [t_1, t_2])} l(v_i, a_j, [t_1, t_2]) \times r(v_i, a_j, [t_1, t_2])$$

The optimal decomposition over the interval $[t_1, t_2]$ is one which minimises the total access cost.

CHARACTERISING SPHERES OF INFLUENCE

We are currently conducting experiments to characterise the spheres of influence in a number of agent-based simulations. In this section, we report the preliminary results of one of these experiments from a simple predator and prey simulation.[2]

The simulation was originally developed to study the effect of different herding or flocking behaviours on predation rates in a population of simulated predators and prey. The simulation consists of a number of predator and prey agents and a toroidal environment containing variable numbers of randomly distributed obstacles

[2]We would like to thank Nick Hawes for providing the code for the simulation.

and food items. Each agent has a number of sensors with differing ranges, which allow the agent to sense objects and other agents within a circular region centred at its current position. For example, predators sense the position of prey and prey sense the location of predators, food and other prey. Each agent also has a collection of basic behaviours. In the case of predators, these include wandering (random motion in the environment), resting and attacking prey.[3] The behaviours of prey agents include wandering, eating food, escaping from predators, and flocking, which causes the prey agents to form groups. The simulation was developed using the SIM_AGENT toolkit (Sloman & Poli 1996), a centralised, time-drive simulator for multi-agent systems.

The predator and prey simulation is typical of many simulations in the multi-agent systems literature and is a useful test case for our approach. It has a large shared state (representing the environment of the agents), and at any given time, each agent accesses only a subset of the state. This subset changes over time, as the agents move to escape predators or to find food. The agent's actions change the environment and hence the behaviour of other agents: when a prey agent eats some food, the food becomes unavailable to other prey which may 'starve' as a consequence. In addition, there is a reasonable probability that at least some of the state variables accessed by one agent will also be accessed by other agents, both because predators tend to follow prey, and because of the flocking of prey.

In the results presented below, we have made a number of simplifying assumptions. The environment is limited to 400 units by 400 units in size and we have considered only the largest sphere of influence for each agent: the state variables representing a circular region of the environment of radius 100 units in the case of a predator, and a smaller region of radius 50 units in the case of the prey. A predator agent accesses approximately 19.6% of the state variables at each timestep (31,417 variables). Of these, between 96.8% and 100% will be accessed by the predator at the next timestep, due to limitations on the speed of the predator. In contrast, a prey agent accesses only 4.9% of the state variables at any timestep, of which between 94.9% and 100% will be accessed at the next timestep.

Table 1 shows the probability that a randomly selected state variable will be accessed by exactly 0, 1, 2 etc. agents at any given timestep for two populations: 1 predator and 5 prey, and 1 predator and 10 prey. Initial placement of the agents, food and obstacles was random, each simulation was run for 100 timesteps and the results have been averaged over 50 runs. As expected, the increased number of agents results in an increase in the probability that any given state variable will be accessed by more than one agent from 0.325 to 0.433. However, the probability of a randomly selected variable being accessed by three or more agents is still relatively small at less than 4%.

Prey	0	1	2	3	4	5
5	0.675	0.257	0.058	0.008	0.001	0
10	0.567	0.282	0.111	0.031	0.007	0.001

Table 1: Probability that a randomly selected state variable will be accessed by n agents.

From the data in table 1, it is apparent that the conditional probability of a state variable being accessed by at least two agents given that it is accessed by at least one is 0.206 in the case of 5 prey and 0.346 in the case of 10 prey.

Prey	1	2	3	4	5
5	0.584	0.211	0.050	0.002	0
10	0.419	0.332	0.104	0.029	0.009

Table 2: Probability that a randomly selected state variable accessed by one prey agent will be accessed by exactly n prey agents.

Further analysis of the pattern of state accesses by prey agents is reported in table 2, which gives the probability that a randomly selected member of the set of state variables accessed by one agent is also accessed by exactly 1, 2, 3 etc. other prey agents. As can be seen, even in the case of 10 prey, the probability of a state variable being shared by three or more agents is still quite small, at less than 15%.

These data suggest that, while significant numbers of variables are shared between agents, relatively few variables are shared by more than three agents. Moreover, each agent accesses a relatively small proportion of the total state at any given time. It therefore suggests that the approach outlined in the previous section, of decomposing the state into a number of CLPs, each responsible for a small number of agents and part of the state may be feasible. However more work is needed to char-

[3] If a predator 'catches' a prey agent, the prey agent is removed from the simulation and the predator gets a food reward.

acterise the stability of the sets of state variables.

SUMMARY

In this paper we have described a framework for the distributed simulation of agent-based systems which aims to overcome some of the deficiencies of the ad-hoc, centralised, time-driven simulation approaches typically employed for agent simulation. Our framework uses the notion of 'spheres of influence' as a basis for dynamically partitioning the shared state of the simulation model into logical processes, and we have described an algorithm for dynamically partitioning the simulation to perform load balancing. We have presented preliminary results of experiments to characterise the spheres of influence in a simple agent-based simulation which suggests that our approach may be feasible.

Future work will include further investigation of the use of spheres of influence for dynamic partitioning and load balancing in a distributed environment, the development of suitable optimistic synchronisation protocols, and ultimately the implementation of a generic simulation kernel.

ACKNOWLEDGEMENTS

Natasha Alechina read an earlier version of this paper and made many helpful comments.

References

Berrached, A., Beheshti, M., Sirisaengtaksin, O. & deKorvin, A. (1998), Alternative approaches to multicast group allocation in hla data distribution, *in* 'Proceedings of the 1998 Spring Simulation Interoperability Workshop'.

Bradshaw, J., ed. (1997), *Software Agents*, AAAI Press, Menlo Park, CA.

Calvin, J. O., Chiang, C. J. & Van Hook, D. J. (1995), Data subscription, *in* 'Proceedings of the Twelfth Workshop on Standards for the Interoperability of Distributed Simulations', pp. 807–813.

Franklin, S. & Graesser, A. (1997), Is it an agent, or just a program?: A taxonomy for autonomous agents, *in* J. Müller, M. J. Wooldridge & N. R. Jennings, eds, 'Intelligent Agents III: Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages', Vol. 1193 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 21–35.

Jennings, N. R. & Wooldridge, M. (1998), Applications of intelligent agents, *in* N. R. Jennings & M. Wooldridge, eds, 'Agent Technology: Foundations, Applications, Markets', Springer-Verlag, pp. 3–28.

Macedonia, M., Zyda, M., Pratt, D. & Barham, P. (1995), Exploiting reality with multicast groups: a network architecture for large-scale virtual environments, *in* 'Virtual Reality Annual International Symposium', pp. 2–10.

Mastaglio, T. W. & Callahan, R. (1995), 'A large-scale complex virtual environment for team training', *IEEE Computer* **28**(7), 49–56.

Morse, K. L. (1996), Interest management in large scale distributed simulations, Technical Report 96-27, Department of Information and Computer Science, University of California, Irvine.

Morse, K. L., Bic, L., Dillencourt, M. & Tsai, K. (1999), Multicast grouping for dynamic data distribution management, *in* 'Proceedings of the 31st Society for Computer Simulation Conference (SCSC '99)'.

Saville, J. (1997), Interest management: Dynamic group multicasting using mobile java policies, *in* 'Proceedings of the 1997 Fall Simulation Interoperability Workshop', number 97F-SIW-020.

Sloman, A. (1998), What's an AI toolkit for?, *in* J. Baxter & B. Logan, eds, 'Software Tools for Developing Agents: Papers from the 1998 Workshop', number Technical Report WS–98–10, AAAI Press, pp. 1–10.

Sloman, A. & Poli, R. (1996), SIM_AGENT: A toolkit for exploring agent designs, *in* M. Wooldridge, J. Mueller & M. Tambe, eds, 'Intelligent Agents II: Agent Theories Architectures and Languages (ATAL-95)', Springer–Verlag, pp. 392–407.

Smith, J., Russo, K. & Schuette, L. (1995), Prototype multicast ip implementation in ModSAF, *in* 'Proceedings of the Twelfth Workshop on Standards for the Interoperability of Distributed Simulations', pp. 175–178.

Steinman, J. S. & Weiland, F. (1994), Parallel proximity detection and the distribution list algorithm, *in* 'Proceedings of the 1994 Workshop on Parallel and Distributed Simulation', pp. 3–11.

Theodoropoulos, G. & Logan, B. (1999), A framework for the distributed simulation of agent-based systems, *in* H. Szczerbicka, ed., 'Modelling and Simulation: a tool for the next millenium, Proceedings of the 13th European Simulation Multiconference (ESM'99)', Vol. 1, SCS, Society for Computer Simulation International, Society for Computer Simulation International, pp. 58–65.

Wooldridge, M. & Jennings, N. R. (1995), 'Intelligent agents: Theory and practice', *Knowledge Engineering Review* **10**(2), 115–152.