An Approach to Interest Management and Dynamic Load Balancing in Distributed Simulation

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

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

KEYWORDS

Distributed simulation, interest management, load balancing, partitioning.

ABSTRACT ***The paper discusses the distributed simulation of systems with large shared state and addresses issues related to interest management and dynamic load balancing. It identifies the efficient partitioning and distribution of the shared state as a key problem in such simulations and outlines a hierarchical multi-level interest management scheme which facilitates dynamic load balancing.***

1 Introduction

Various approaches for exploiting parallelism at different levels in simulation problems have been developed (Ferscha & Tripathi 1994). 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 timestamped events in the simulation. From an LP's point of view, two types of events are distinguished; namely internal events which have a causal impact only on the state variables of the LP, and external events which may also have an impact on the states of other LPs. External events are typically modelled as timestamped messages exchanged between the LPs involved. The purpose of this interaction is to exchange information regarding the values of the state variables which are of common interest to the LPs involved in the communication (the shared state).

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 systems which operate in a complex environment and interact with it in complex and dynamic patterns (such as multi-agent systems, battlefield simulations, ecological systems, games etc.), 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. Which variables the processes do in fact access depends both on the state of the process and the state of the processes simulating the environment. Encapsulating the shared state in a single process (e.g., via some centralised scheme) introduces a bottleneck, while distributing it all across the LPs (in a decentralised, event driven scheme) will typically result in frequent all-to-all communication and broadcasting, which is extremely costly and results in the loss of many of the advantages of distributed simulation.

Therefore, what is required is an alternative approach to decompose and distribute the shared state, which minimises bottlenecks and broadcast communication and by implication, maximises performance. Furthermore, the dynamically changing interaction patterns between the constituent parts of the simulated system and between the system and its environment call for the dynamic reconfiguration of the simulation to balance the load and sustain high performance.

This paper, which summarises our previous work on distributed simulation of multi-agent systems, outlines

a conceptual unified framework which supports the efficient decomposition and distribution of the shared state and facilitates load balancing. Issues addressed in this paper have also been discussed in more detail in (Theodoropoulos & Logan 1999*b*, Theodoropoulos & Logan 1999*a*, Logan & Theodoropoulos 2000, Logan & Theodoropoulos n.d.).

2 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). The data of interest to a process is referred to as its *Domain of Interest* (DOI). Special entities in the simulation, referred to as *Interest Managers*, are responsible for filtering generated data and forwarding it to the interested processes based on their IEs (Morse 1996). The region of the multi-dimensional parameter space in which an Interest Manager is responsible for managing data transmission is referred to as its *Domain of Responsibility* (DOR).

Various Interest Management schemes have been devised, utilising different communication models and filtering schemes. 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 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). For example, the High Level Architecture (HLA) 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 for publishing data (update regions) (Def 1998). In existing HLA 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 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 and are therefore incapable of balancing the communication and computational load, with the result that performance is often poor. 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. Some early systems, such as JPSD (Macedonia et al. 1995) and STOW-E (Van Hook, Calvin, Newton & Fusco 1994) did exhibit some degree of dynamism in their filtering schemes. 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 example, Berrached et al. (Berrached, Beheshti, Sirisaengtaksin & de Korvin 1998) examine hierarchical grid implementations and a hybrid grid/clustering scheme of update regions to dynamically reconfigure multicast groups while Morse et al. (Morse, Bic, Dillencourt & Tsai 1999) report on preliminary investigations of an 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. The Joint MEASURE system (Hall, Zeigler & Sarjoughian 1999, Hall 2000, Sarjoughian, Zeigler & Hall 2000) is implemented on top of HLA and utilises event distribution and predictive encounter controllers to efficiently manage interactions among entities. However, despite these efforts, the problem of dynamic interest management remains largely unsolved.

3 Load Balancing

The synchronisation mechanisms involved in distributed simulation render load balancing techniques developed for other, more conventional classes of parallel applications insufficient. For instance, in the case of optimistic synchronisation, high processor utilisation does not necessarily imply good performance as operations could later be undone (rollback), while process migration can affect the efficiency of the synchronisation mechanism (e.g., amount of roll backed computation). As a result, load balancing has been studied extensively in the special context of both conservative and optimistic parallel simulation (Burdorf & Marti 1993, Glazer & Tropper 1993, Goldberg 1992, Reiher & Jefferson 1990, Schlagenhaft, Ruhwandl, Sporrer & Bauer 1995, Carothers & Fujimoto 1996).

However, the issue of dynamic load balancing has received very little attention in relation to interest management and work in this area to date is only preliminary (Morse 1996, Messina, Davis, Brunette, Gottshock, Curkendall, Ekroot, Miller, Plesea, Craymer, Siegel, Lawson, Fusco & Owen 1997,

White & Myjak 1998, Myjak, Sharp, Shu, Riehl, Berkley, Nguyen, Camplin & Roche 1999).

In the next section we present a new unified framework for dynamic interest management and load balancing.

4 A New Approach

Our approach is based on the notion of *spheres of Influence*, which are used to dynamically decompose and distribute the shared state so that bottlenecks and broadcast communication are minimised. It utilises a dynamic, multi-level, hierarchical filtering scheme which is not confined to grids and rectangular regions of multi-dimensional parameter space nor does it rely on the support provided by the TCP/IP protocols. Furthermore, our approach aims to exploit this decomposition in order to perform dynamic load balancing.

4.1 Spheres of Influence

We assume that each Logical Process generates and responds to at most a finite number of event types. Different types of events will typically have different effects on other LPs, 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 LP which generated the event (e.g., its position in space) 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 LPs in response to the event.

We can use the spheres of influence of the events generated by each LP to derive an idealised decomposition of the shared state. We define the sphere of influence of an LP, p_i over the time interval $[t_1, t_2]$, $s(p_i, [t_1, t_2])$, as the union of the spheres of influence of the events generated by the process over the interval.

Intersecting the spheres of influence for each event generated by the process gives a partial order over sets of state variables for the process 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.

Intersecting the spheres of influence for each process 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 processes. This partial order can be seen as a measure of the difficulty of associating variables with a particular process: the state variables which are members of the sets which are first in the order are required by the largest number of processes, whereas those sets of state variables which come last are required by only a single process.

Any approach to the decomposition and distribution of the shared state 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 processes 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.

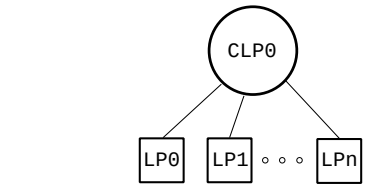
We are currently conducting experiments to characterise the spheres of influence in a number of simulations of agent-based systems. Out preliminary results suggest that the proposed approach feasible. For more detailed information and quantitative results the reader is referred to (Logan & Theodoropoulos n.d.)

4.2 Dynamic State Distribution and Load Balancing

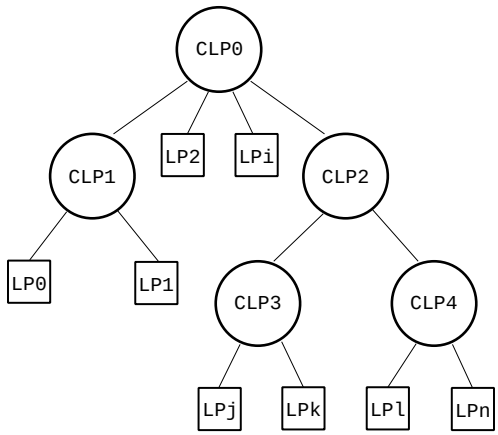
The decomposition of the state is achieved by means of an additional set of Logical Processes, namely *Communication Logical Processes* (*CLPs*). The CLPs act as Interest Managers. Each CLP maintains a subset of the state variables and the interaction of ALPs and ELPs is via the variables maintained by the CLPs. CLPs enable the clustering of LPs 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 LPs in the simulation. Thus, the number and distribution of CLPs is not fixed, but varies during the simulation.

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 LPs are directed to this single CLP, as is all inter-process 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 of the partial order over the spheres of influence. Those groups of LPs whose events and actions have formulated the new CLP(s) communi-



(a)



(b)

Figure 1: Generating the tree of CLPs.

cate 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 LPs are the leaves and the CLPs the intermediate nodes of the tree. Events by the LPs 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 LPs and CLPs and to its parent CLP.

The rank of a variable v_j for process p_i over the interval $[t_1, t_2]$, $r(v_j, p_i, [t_1, t_2])$ is the number of events in whose sphere of influence v_j lies. We define the cost of accessing a variable v_j for a logical process p_i as the rank of v_j for p_i, $r(v_j, p_i)$, times the number of CLPs which must be traversed to reach v_j during the interval $[t_1, t_2]$, $l(v_j, p_i)$, i.e., the cost of accessing variables in the local CLP is 0. Then the cost to an LP p_i of accessing all the variables in its sphere of influence $s(p_i)$ is:

$$\sum_{v_j \in s(p_i)} l(v_j, p_i) \times r(v_j, p_i)$$

and the total access cost for all LPs $p_1, \ldots, p_n$ of a particular decomposition over the interval $[t_1, t_2]$ is:

$$\sum_{i=1}^{n} \sum_{v_j \in s(p_i)} l(v_j, p_i) \times r(v_j, p_i)$$

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

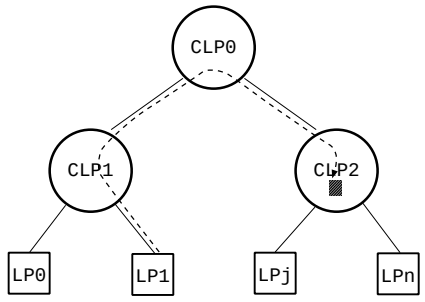
As the total number and distribution of instances of each event type generated by an LP varies, so the partial order over the spheres of influence changes, and the structure of the tree must change accordingly to reflect the LPs' current behaviour and keep the communication and computational load balanced. This may be achieved in two ways, namely by changing the position of the LP in the tree, or by relocating state in the tree. State may be relocated either by moving subsets of the state variables from one CLP to another, or by merging CLPs upwards and then (possibly) splitting them again in a different way.

For example, Figures 2(a) and 2(b) illustrate the migration of LP1 in the tree, to bring it closer to the part of the state it most frequently accesses (denoted by the shaded area in CLP2). If this reduces the load handled by CLP1 sufficiently, it can be merged with CLP0, as depicted in Figure 2(c). Alternatively, the subset of state variables accessed by LP1 in CLP2 could have been moved to CLP1.

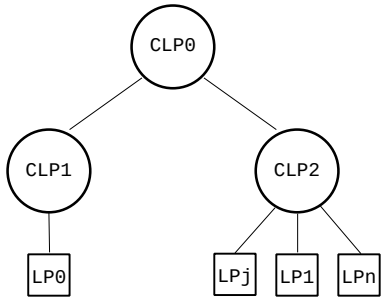
5 Open Issues

A number of challenging issues have to be addressed before this approach is realised. Techniques are required to obtain global snapshots of the distributed simulation and approximate the spheres of influence at any instant, e.g. (Chandy & Lamport 1985, Babaoglou & Marzullo 1993). Furthermore, algorithms for redistributing the state and reorganising the tree to approximate the spheres of influence and balance the load to achieve high simulation performance must be developed; in this context appropriate performance metrics and cost functions need to be defined which will take into account all relevant characteristics of both the host platform (e.g. network configuration, CPU and memory architecture etc.) and the dynamics of the simulated systems (e.g. frequency of interactions and state accesses etc.). To this end, a range of alternative solutions may be envisioned, from the periodic redistribution of the whole state and construction of the tree from scratch to the gradual moving of LPs and state variables through different levels in the dynamically reconfigured tree.

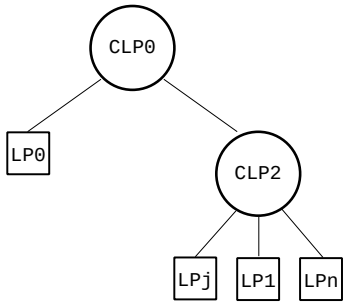
CLPs should be able to respond to various events/queries issued by the LPs regarding shared state. As the state information required to respond to these may be distributed through the tree, appropriate routing algorithms are needed to enable the CLPs to locate this information; this is clearly highly non-trivial.



(a)



(b)



(c)

Figure 2: LP migration and merging of CLPs.

The proposed framework in intended to support both conservative and optimistic synchronisation protocols. To date, most work on Interest Management has been carried out within the context of large-scale, real-time simulations where synchronisation is straightforward, as at any instant, all processes (federates) are approximately at the same wall-clock time. In logical time simulations however, where different LPs will typically be at different logical times, interest management can introduce temporal coherency errors in the simulation (e.g., processes do not receive messages they 'should' have received). Thus far, only a limited amount of work has been done in this area, mainly for HLA and similar grid-based filtering schemes (Tacic & Fujimoto 1998, Steinman & Weiland 1994).

In (Theodoropoulos & Logan 1999*a*, Theodoropoulos & Logan 1999*b*, Logan & Theodoropoulos 2000) we have outlined possible solutions to some of the above issues, however more work is needed to address all these problems.

6 Summary

In this paper we have addressed issues related to the distributed simulation of systems with a large state space shared between their constituent parts. The efficient partitioning of this shared state is a key problem which calls for new and innovative interest management and load balancing schemes. We have described an approach to hierarchical, multi-level dynamic interest management which 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. This is work in progress and we have identified a number of challenging issues which have to be addressed before our approach is realised.

References

Babaoglou, O. & Marzullo, K. (1993), Consistent global states of distributed systems: Fundamental concepts and mechanisms, Technical Report UBLCS-93-1, Laboratory for Computer Science, University of Bologna.

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

Burdorf, C. & Marti, J. (1993), 'Load balancing strategies for Time Warp on multi-user workstations', *The Computer Journal* **36**(2), 168–176.

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.

Carothers, C. & Fujimoto, R. (1996), Background execution of Time-Warp programs, *in* 'Proceedings of 10th Workshop on Parallel and Distributed Simulation', Society for Computer Simulation, Society for Computer Simulation.

Chandy, K. M. & Lamport, L. (1985), 'Distributed snapshots: Determining global states of distributed systems', *ACM Transactions on Computer Systems* **3**(1), 63–75.

Def (1998), *High Level Architecture RTI Interface Specification, Version 1.3*.

Ferscha, A. & Tripathi, S. K. (1994), Parallel and distributed simulation of discrete event systems, Technical Report CS.TR.3336, University of Maryland.

Glazer, D. W. & Tropper, C. (1993), 'On process migration and load balancing in Time-Warp', *IEEE Transactions on Parallel and Distributed Systems* **3**(4), 318–327.

Goldberg, A. (1992), Virtual time synchronisation of replicated processes, *in* 'Proceedings of 6th Workshop on Parallel and Distributed Simulation', Society for Computer Simulation, Society for Computer Simulation, pp. 107–116.

Hall, S. B. (2000), Using Joint MEASURE to study tradeoffs between network traffic reduction and fidelity of HLA compliant pursuer/evader simulations, *in* 'Proceedings of the Summer Simulation Conference', Society for Computer Simulation, Vancouver, Canada.

Hall, S. B., Zeigler, B. P. & Sarjoughian, H. (1999), Joint MEASURE: Distributed simulation issues in a mission effectiveness analytic simulator, *in* 'Proceedings of the Simulation Interoperability Workshop', Orlando, FL.

Logan, B. & Theodoropoulos, G. (2000), Dynamic interest management in the distributed simulation of agent-based systems, *in* J. S. Sarjoughian, F. E. Cellier, M. M. Marefat & J. W. Rozenblit, eds, 'Proceedings of the Tenth Conference on AI, Simulation and Planning, AIS-2000', Society for Computer Simulation International, pp. 45–50.

Logan, B. & Theodoropoulos, G. (n.d.), 'The distributed simulation of multi-agent systems', *Proceedings of the IEEE*.

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.

Messina, P., Davis, D., Brunette, S., Gottshock, T., Curkendall, D., Ekroot, L., Miller, C., Plesea, L., Craymer, L., Siegel, H., Lawson, C., Fusco, D. & Owen, W. (1997), Synthetic forces express: A new initiative in scalable computing for military simulation, *in* 'Proceedings of the 1997 Spring Simulation Interoperability Workshop', IST.

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)'.

Myjak, M., Sharp, S., Shu, W., Riehl, J., Berkley, D., Nguyen, P., Camplin, S. & Roche, M. (1999), Implementing object transfer in the HLA, Technical report.

Reiher, P. L. & Jefferson, D. (1990), 'Dynamic load management in the Time-Warp operating system', *Transactions of the Society for Computer Simulation* **7**(2), 91–120.

Sarjoughian, H. S., Zeigler, B. P. & Hall, S. B. (2000), 'A layered modeling and simulation architecture for agent-based system development', *Proceedings of the IEEE*.

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.

Schlagenhaft, R., Ruhwandl, M., Sporrer, C. & Bauer, H. (1995), Dynamic load balancing of a multi-cluster simulation on a network of workstations, *in* 'Proceedings of 9th Workshop on Parallel and Distributed Simulation', Society for Computer Simulation, Society for Computer Simulation, pp. 175–180.

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.

Tacic, I. & Fujimoto, R. (1998), Synchronised data distribution management in distributed simulations, *in* 'Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS98)', pp. 108–115.

Theodoropoulos, G. & Logan, B. (1999*a*), Distributed simulation of agent-based systems, *in* B. H. V. Topping, ed., 'Developments in Computational Mechanics with High Performance Computing: Proceedings of the Third Euro-conference on Parallel and Distributed Computing for Computational Mechanics', Civil-Comp Press, pp. 147–154.

Theodoropoulos, G. & Logan, B. (1999*b*), 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.

Van Hook, D., Calvin, J., Newton, M. & Fusco, D. (1994), An approach to DIS scaleability, *in* 'Proceedings of the 11th Workshop on Standards for the Interoperability of Distributed Simulations', pp. 347–356.

White, E. & Myjak, M. (1998), A conceptual model for simulation load balancing, *in* 'Proceedings of the 1998 Spring Simulation Interoperability Workshop'.

Author Biographies

GEORGIOS THEODOROPOULOS received a Diploma degree in Computer Engineering from the University of Patras, Greece in 1989 and MSc and PhD degrees in Computer Science from the University of Manchester, U.K. in 1991 and 1995 respectively. Since February 1998 he has been a Lecturer in the School of Computer Science, University of Birmingham, U.K. teaching courses on Hardware Engineering and Computer Networks. His research interests include parallel and distributed systems, computer and network architectures and modelling and distributed simulation.

BRIAN LOGAN is a lecturer in the School of Computer Science and IT at the University of Nottingham,

UK. He received a PhD in design theory from the University of Strathclyde, UK in 1986. His research interests include the specification, design and implementation of agent-based systems, including logics and ontologies for agent-based systems and software tools for building agents. Before moving to Nottingham, he was a member of the Cognition and Affect group at the University of Birmingham, where he worked on agent-related projects funded by the UK Defence Evaluation and Research Agency and the Leverhulme Trust, developing architectures for autonomous intelligent agents capable of complex decision making under constraints such as incomplete and uncertain information and time pressure.