Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

DEVELOPING INTEREST MANAGEMENT TECHNIQUES IN DISTRIBUTED INTERACTIVE SIMULATION USING JAVA

Simon J.E. Taylor
Jon Saville
Rajeev Sudra

Centre for Applied Simulation Modelling
Department of Information System and Computing
Brunel University, Uxbridge
UB8 3PH, UNITED KINGDOM

ABSTRACT

Bandwidth consumption in distributed real-time simulation, or networked real-time simulation, is a major problem as the number of participants and the sophistication of joint simulation exercises grow in size. This paper briefly reviews distributed real-time simulation and bandwidth reduction techniques and introduces the Generic Runtime Infrastructure for Distributed Simulation (GRIDS) as a research architecture for studying such problems. GRIDS uses Java abstract classes to promote distributed services called thin agents, a novel approach to implementing distributed simulation services, such as user-defined bandwidth reduction mechanisms, and to distributing the executable code across the simulation. Thin agents offer the advantages of traditional agents without the overhead imposed by mobility or continuous state, which are unnecessary in this context. We present our implementation and some predicted results from message-reduction studies using thin agents.

1 INTRODUCTION

Since the initial success of the distributed military simulation effort SIMNET (Miller and Thorpe 1995), the use of distributed real time simulation (sometimes called networked real time simulation) for the purpose of training has grown over the past decade. Perhaps one of the most well known examples of distributed real time simulation is Distributed Interactive Simulation (DIS) (IEEE 1994). DIS aims to provide a high performance virtual training environment by systematically connecting separate simulators located at geographically distributed sites (STRICOM 1992). The training environment is virtual in the sense of real people operating in simulated systems. The key elements in such a system are the simulators and the connecting network (usually a mix of LAN and WAN technology). Page et al. (1997) review the Joint Training Confederation, one of the largest applications of DIS-like technology. High Level Architecture (DMSO 1996) represents the most recent and significant advances in distributed real-time simulation technologies.

One of the major problems in distributed real-time simulations is bandwidth consumption. Consider an illustrative example, a training scenario where command staff in different locations are engaged in a wargame (although other applications such as air traffic control, emergency/evacuation training, traffic control systems, etc. are possible). Each site has semi-automated force (SAF) simulators, representing the combatants, and command port simulators used by the trainees as the interface to the wargame. The command post simulators will typically include tactical displays showing the disposition of forces. A LAN carries state information between simulation entities at each site, and a WAN carries data (with attendant time latency) between sites. Each simulation entity must regularly broadcast state information to ensure that tactical displays are updated within the hard deadlines imposed by real time operation. Unfortunately, this scenario does not scale well. As the number of simulation entities increases, network bandwidth is quickly consumed by state information that is irrelevant to the majority of entities taking part. The situation deteriorates when manned vehicle simulators are added; these require out-of-the-window displays with low update latency to preserve the illusion of reality.

The paper is structured as follows. In section 2 we review techniques that have been used to reduce the volume of network traffic between simulators. In section 3 we present the Generic Runtime Infrastructure for Distributed Simulation (GRIDS), our contribution to the field of real time simulation, and discuss how these might be used to reduce network traffic. Section 4 gives a case study to illustrate distributed services and their

performance. The paper ends with some conclusions in Section 5.

2 BANDWIDTH REDUCTION TECHNIQUES

We recognize that bandwidth-reduction techniques fall into two broad categories—data aware and data independent. The former refers to intelligent marshaling techniques that direct information to interested parties. The latter, including packet bundling and data compression, apply gross techniques to the general data flow, and do not scale well. Bassiouni et al. (1997) give an excellent review of some of these bandwidth techniques. Our work focuses on data-aware techniques, which we briefly review.

Simple approach. The originating entity determines the set of entities which may be interested in a particular piece of data, and replicates separate messages to each target. The infrastructure provides no assistance in identifying interested parties and merely delivers the messages.

Dead reckoning. In a simulation employing dead reckoning, each process maintains a movement model (or dead reckoning algorithm) for every other process. The model estimates future movements based on past behavior, and is updated using infrequent broadcast messages from the process being modeled.

Network subscription. Targets choose to listen to network broadcast channels which carry information of relevance. The TCP/IP suite provides this facility in the form of IP multicasting. Processes broadcast classes of data (for example, state information affecting ground-based vehicles) on the relevant channel.

Group subscription. This model, employed by the PVM library (Geist et al. 1994), requires each target to join a group. Messages sent to that group are duplicated by the infrastructure and dispatched to all members. This model has been greatly extended by the MPI standard (MPIF 1995), which allows group membership to be manipulated in complex ways.

Relevance filtering. These schemes depend on the infrastructure to direct state information to interested parties. Such a scheme is exemplified by the HLA publish/subscribe model of data distribution, which provides two mechanisms. Filtering on the basis of an object's class is performed by the declaration management services. Filtering based on attribute value is performed by the data distribution management services, in the form of routing spaces (Morse and Steinman 1997).

These techniques reduce bandwidth either by managing the transmission of messages to interested parties only, or by employing algorithms which reduce the frequency of state updated. Individually these can be useful in specific circumstances. For example, network subscription is a valid approach when the number of classes of state update is small. Dead reckoning is appropriate when simulation entities are slow-moving or network bandwidth is relatively high.

However, the designer of a new simulation must plan the implementation based on the availability of support mechanisms in the communication infrastructure, since few runtime systems support all techniques. The alternative, creating a bespoke solution, is costly.

We suggest that an infrastructure could support a variety of bandwidth reduction techniques by providing a mechanism which allows simple decision-making units to be distributed and executed in support of those techniques. We propose a novel code distribution paradigm, called *thin agents*, which we believe offers this ability at an appropriate level of abstraction.

3 THIN AGENTS

3.1 Definition

Thin agents are the basis on which distributed simulation services that we wish to provide are realised. They share many characteristics with traditional agents (Jennings, Sycara and Wooldrige 1998, White 1994). In both cases a code fragment or algorithm is compiled into executable form. Rather than being permanently located at a single node, thin and traditional agents can move through the infrastructure.

A traditional agent requires continuous state and the ability to move between nodes to accomplish its task, under its own motivation. In contrast, a thin agent is invoked where needed and thus has minimal housekeeping code overhead. We use the term *thin* to denote minimum functionality from which larger, more complex agents can be built (see for example Silva and Delgado 1998).

3.2 Thin Agent Support Requirements

Thin agents require a small set of services from the infrastructure. It must:

- distribute copies of the thin agent to each simulation node, ready for instantiation. In this way the functionality provided by the thin agent will be available at wherever it is required.
- provide an execution context in which to instantiate the thin agent. The infrastructure must be able to invoke the functionality encapsulated by the thin agent in a safe environment.
- allow the thin agent access to the basic communication services of the infrastructure. The mechanisms which thin agents encapsulate may need access to the state of the simulation in order to perform their bandwidth reduction role.

4 IMPLEMENTATION

We have implemented the thin agent scheme described and an enabling infrastructure supporting a broad range of simulation types. Thin agents have been used in a variety of distributed simulation projects which have underlined their role in bandwidth reduction. Our infrastructure is called GRIDS (Generic Runtime Infrastructure for Distributed Simulation) (Saville and Taylor 1997).

GRIDS is built using the Java language (Arnold and Gosling 1996). Java was chosen since it offers a large set of features appropriate to distributed system construction. In particular, Java provides a safe environment to invoke objects loaded from across the network, a model directly supporting our thin agent scheme.

GRIDS consists of a set of communicating servers. An entity can join an executing simulation by connecting to a server. GRIDS provides several message facilities and the ability to publish attributes which can be queried by other entities (via a metadatabase). Figure 1 shows the GRIDS protocol architecture and Figure 2 shows the client-server connectivity that GRIDS supports.

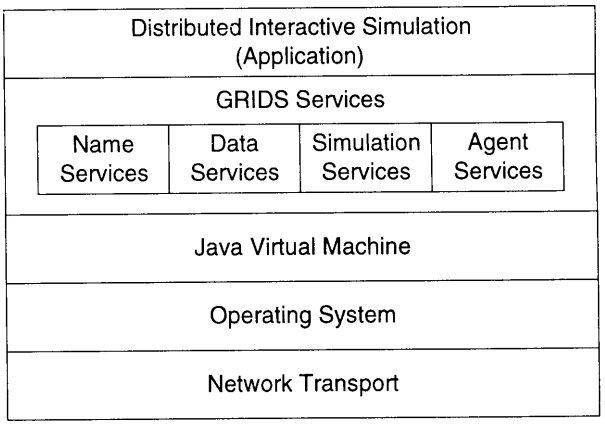


Figure 1: GRIDS Protocol Diagram

A thin agent, encapsulating a bandwidth reduction mechanism, consists of a Java class which is uploaded to the infrastructure by the publishing simulation entity. GRIDS replicates the class to every simulation node. When the mechanism is required by a simulation entity (not necessarily the publishing entity) the GRIDS server local to the requester instantiates the thin agent and invokes its functionality. This mechanism is shown in Figure 3.

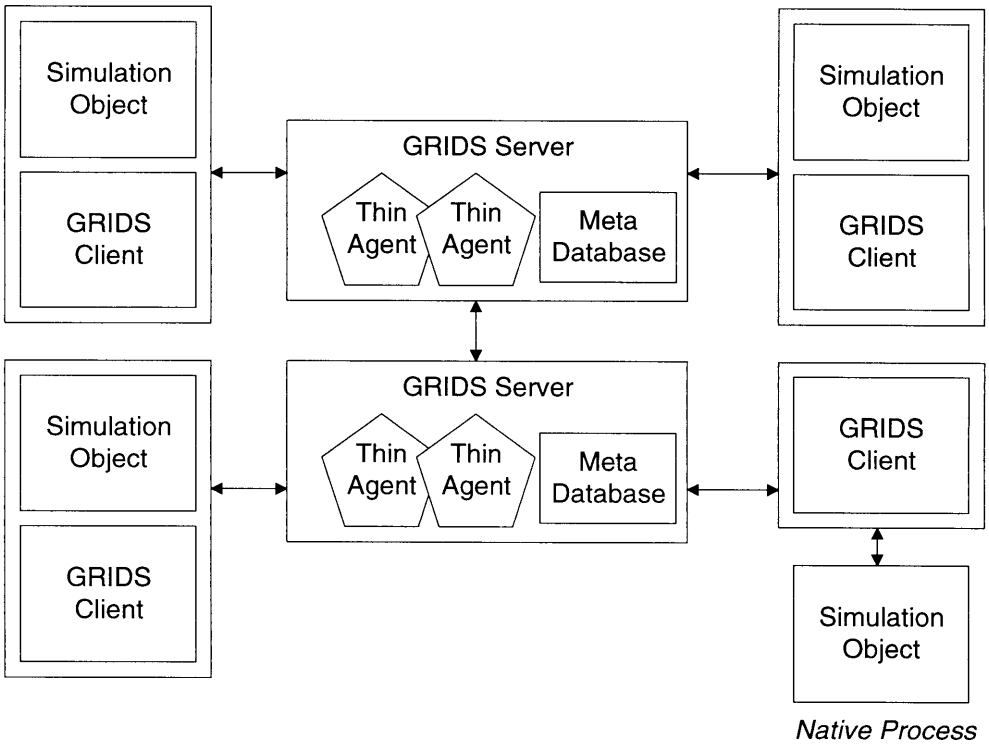


Figure 2: GRIDS Architecture

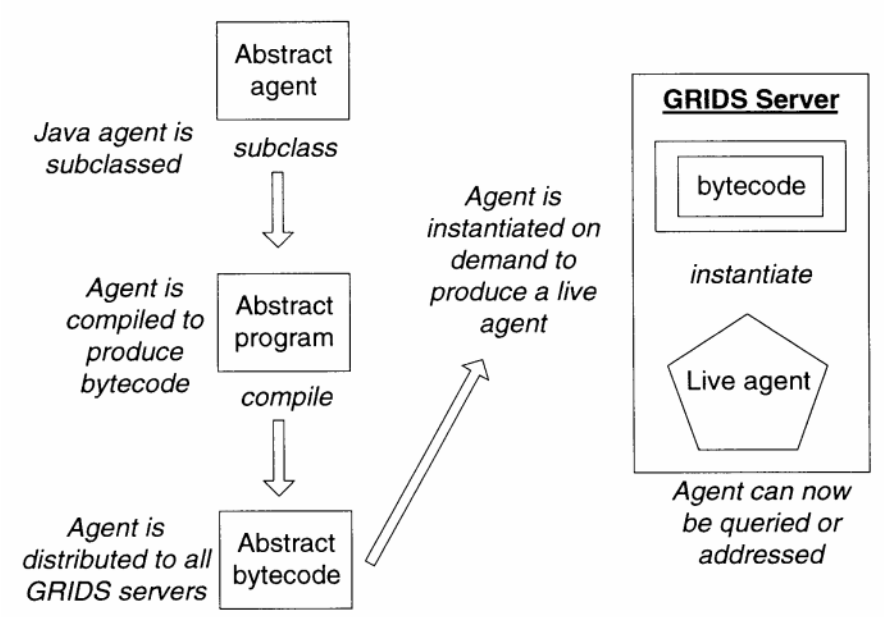


Figure 3: Thin Agent Mechanism

In the context of GRIDS, thin agents perform two major functions, described below. It should be noted that thin agents are not limited to these uses; they represent merely the first application of this approach.

Attributes. A thin agent can be logically attached to an attribute. When the attribute value is requested by an entity the thin agent is invoked and queried for an updated value. A thin agent maintains state at the GRIDS server where it was invoked, and thus can encapsulate services such as dead reckoning.

Group multicasting. The second major use of thin agents in GRIDS is as part of the GRIDS group multicasting facility. A group is a logical target for messages, and represents a set of recipients. In contrast to PVM groups, where recipients have to manually subscribe, a GRIDS group has dynamic membership. A message directed to the group causes the local (to the sender) GRIDS server to instantiate the associated thin agent, which returns a list of recipients. The thin agent can thus perform message filtering using the arbitrary user-defined algorithm it is carrying.

4 CASE STUDY

The case study discussed here focuses on the ability of thin agents to reduce bandwidth. Specifically, thin agents are used to reduce the number of message exchanges required in a simple distributed simulation.

The scenario consists of a variable number of entities representing tanks. Each tank has a randomly assigned waypoint, toward which it travels at a constant speed. If a tank detects another tank within range it pauses to fire a shell. Tanks that are hit retire from the simulation.

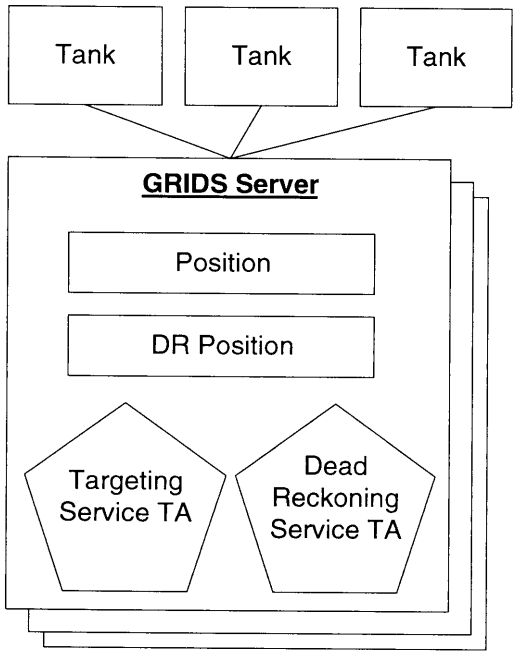
The set of objects involved is illustrated in Figure 4. Each tank updates the value of attribute POSITION every 10 seconds. In order that other entities receive accurate position values, a thin agent encapsulates a simple dead-reckoning algorithm which calculates a tank's likely position based on its past behavior. Entities access these estimates by querying the attribute DR POSITION, to which the thin agent is attached.

A tank obtains a list of threats (other tanks within firing range) by directing a message towards the THREAT REFLECTOR group. When the messages reaches the GRIDS server the thin agent attached to that group is invoked. The thin agent queries the DR POSITION attribute of the remaining tanks and returns a message to the inquiring object which contains a list of threats.

In this scenario the use of a dead-reckoning algorithm reduces the number of position update messages required from each tank by interpolating past behavior. The thin agent approach offers additional advantage over a traditional DIS-type dead reckoning scheme. Firstly, rather than each simulation entity maintaining dead-reckoning models for every other simulation entity, only one model for each tank is maintained by the GRIDS servers. Secondly, the dead-reckoning model can be changed, and additional entities modeled, without rebuilding every other simulation object.

Similarly, the thin agent which identifies potential threats eliminates the requirement for every tank object to perform this functionality individually.

Figure 5 shows the predicted rate of message exchange between tank objects, before firing commences. The dotted line shows results obtained with no bandwidth reduction –



Replicated at each server

Figure 4: Case Study Example

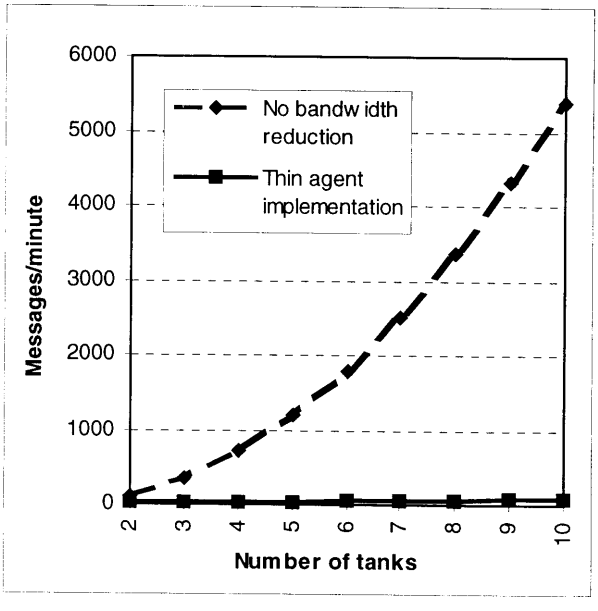


Figure 5: Predicted Results

the number of messages increases geometrically as tank objects are added to the simulation. The solid line shows the effects of using thin agents to eliminate unnecessary message exchanges. The message growth rate is linear with the number of tanks.

This communication saving is not without cost. The invocation of each thin agent requires processing resources at each GRIDS server. An overall benefit is realized since the savings accrued from reduced communication usually outweigh the additional processing requirements.

5 CONCLUSIONS

This paper has reviewed several techniques of bandwidth reduction for distributed interactive simulation. We have introduced a novel paradigm for implementing bandwidth reduction mechanisms using mobile objects under the GRIDS system. This work is presented as a possibly interesting research architecture that exploits several Java features in the hope of providing services for distributed real-time simulation. It is in no way meant to compete with the High Level Architecture that has been shown to successfully implement large distributed interactive simulations. This contribution merely attempts investigate possible alternatives for smaller server-based systems. Another note is the runtime execution speed of Java. Again this work is presented as investigative research rather than development for a commercial product. It is our intention to prototype possible architectures. If this kind of architecture was to meet with success then it is possible that a high speed implementation could be implemented in C++ or possibly future versions of Java (with optimisation). At the moment it is enough that theory suggests adequate performance.

This work is continuing through further development of the infrastructure and its applicability to problem domains other than DIS (web-based simulation and the role of Java in discrete event simulation (Odhabi, Paul and Macredie 1998). Examples of the GRIDS implementation are available from: www.brunel.ac.uk/research/casm.

REFERENCES

Arnold, K. and J. Gosling. 1996. The Java Programming Language. Addison-Wesley, USA.

Bassiouni, M.A., M.-H. Chiu, M. Loper, M.Garnsey, and J. Williams. 1997. Performance and Reliability Analysis of Relevance Filtering for Scalable Distributed Interactive Simulation. *ACM Transactions on Modeling and Computer Simulation*, 7 (3) (July): 293-331.

DMSO 1996. HLA Rules, Version 1.0, August 15, 1996. *http://www.dmso.mil/projects/ hla*.

Geist, A; A. Beguelin; J. Dongarra; W. Jiang; R. Manchek; and V. Sunderam. 1994, *PVM: Parallel Virtual Machine*. MIT Press, USA.

IEEE 1278-1993. 1994. Standard for Information Technology - Protocols for Distributed Interactive Simulation Applications IEEE Press, USA.

Jennings, N., K. Sycara and M Wooldridge. 1998. "A Roadmap of Agent Research and Development." *Autonomous Agents and Mult-Agent Systems*, 1:275-306.

Message Passing Interface Forum (MPIF). 1995. MPI: A Message Passing Interface Standard, Version 1.1., Web address http://www.ececs.uc.edu/miscdocs/mpi-report-1.1/mpi-report.html

Morse, K. and J. Steinman. 1997, Data Distribution Management in the HLA: Multidimensional Regions and Physically Correct Filtering. In *Proceedings of the 1997 Spring Software Interoperability Workshop*. Orlando, FL, USA.

Miller, D. C. and J. A. Thorpe. 1995. SIMNET: The advent of simulator networking. *Proceedings of the IEEE*, 83 (8):1114-1123.

Page E.H.; B.S. Canova; and J.A. Tufarolo 1997. A Case Study of Verification, Validation and Accreditation for Advanced Distributed Simulation. *ACM Transactions on Modeling and Computer Simulation*, 7 (3) (July): 393-424.

Saville, J. and S.J.E Taylor. 1997. Interest management: Dynamic group multicasting using mobile Java policies. in *Proceedings of the Fall 1997 Software Interoperability Workshop*. Orlando, FL, USA.

Silva, A. and J. Delgado. 1998. The Agent Pattern for Mobile Agent Systems. In *Proceedings of the 1998 European Conference on Pattern Languages of Programming and Computing*. Available from berlin.inesc.pt.pt/alb/publications.htm

STRICOM. 1992. *Distributed Interactive Simulation – Operational Concepts Draft 2.1*. September.

White, J. 1994. *Telescript technology: The foundation for the electronic marketplace*. General Magic, Inc.

AUTHOR BIOGRAPHIES

SIMON J.E. TAYLOR is the Chair of the Simulation Study Group of the UK Operational Research Society. He is a lecturer in the Department of Information Systems and Computing and is a member of the Centre for Applied Simulation Modelling, both at Brunel University, UK. He was previously part of the Centre for Parallel Computing at the University of Westminster. He has an undergraduate degree in Industrial Studies (Sheffield Hallam), a M.Sc. in Computing Studies (Sheffield Hallam) and a Ph.D. in Parallel and Distributed Simulation (Leeds Metropolitan). His main research interests are distributed simulation and applications of simulation health care. He has also been known to occasionally tread the boards.

JON SAVILLE is a Ph.D. candidate in the Department of Information Systems and Computing at Brunel University, UK. He has worked in the telecommunications industry for many years developing communications software infrastructure. Recently his research has focused on methods of constructing large distributed systems from components primitives.

RAJEEV SUDRA is a Ph.D. candidate in the Department of Information Systems and Computing at Brunel University, UK. He received his B.Sc. in Computer Science and Economics also from Brunel University. He has gained much experience working in industry ranging from distributed systems software development to designing and deploying large-scale computer networks. His research focuses on real-time simulation and agent-based systems.