Adaptive Message Clustering for Distributed Agent-Based Systems

Cole Sherer
Abhishek Gupta
Maria Hybinette

Computer Science Department
University of Georgia
Athens, GA 30602, USA

***Abstract*—Many agent-based simulation kernels rely on message passing in their core implementation. As the number of agents in a simulation increases or as the complexity of their communication expands the number of messages can increase exponentially. This is troublesome because the message content itself may be quite small, while the overhead, including message headers can dominate bandwidth and processing time. In these cases message passing becomes a bottleneck to scalability. The overhead of message exchange may saturate the network and degrade performance of the simulation. One approach to this challenge that has been investigated in related networking and simulation research centers is combining or "piggy-backing" multiple small messages together with a consolidated header. In many applications performance improves as larger, but fewer messages are sent. However, the pattern of message passing is different in the case of agent-based simulation (ABS), and this approach has not yet been explored for ABS systems.**

In this work we provide an overview of the design and implementation of a message piggy-backing approach for ABS systems using the SASSY platform. SASSY is a hybrid, large-scale distributed ABS system that provides an agent-based API on top of a PDES kernel. We provide a comparative performance evaluation for implementations in SASSY with a combined RMI and shared memory message passing approach. We also show performance of our new adaptive message clustering mechanism that clusters messages when advantageous and avoids clustering when the overhead of clustering dominates.

INDEX WORDS: Agent-Based Simulation, Distributed Simulation, Piggy-backing, Message Clustering, SASSY

I. Introduction

Agent-based simulation systems have gained significance in recent years because they provide a more natural way for developers to design their simulations than traditional discrete event simulators. ABS systems are applied in the study of multi-robot systems, social animal behavior, automobile traffic, and many other fields. The kernel of an ABS system implements communication, and a sense-think-act cycle using message passing. Because message passing is central to an ABS system, it may also become a bottleneck. The efficiency of this message passing is the focus of this research.

The topology of communication between agents depends on the simulation scenario. Some simulations may have agents with a limited sensing range, for instance ants have a limited sensing range in comparison to the size of an entire simulation environment (see Figure 1). In this case, the communication topology is simplified significantly from the fully connected case, which is often assumed in simulation implementations. Our approach is to exploit limitations to communication observed in real systems, such as limited sensing range or obstructions in terrain.

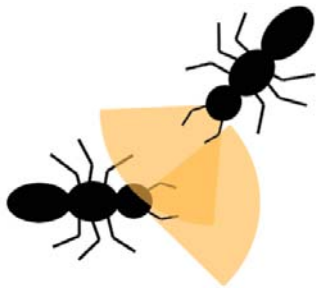

Fig. 1. Limited Sensing Range of Ants

As the complexity of ABS scenarios increase, through more agents or more communication, the number of messages exchanged may increase exponentially. This may degrade performance and result in a significant increase in execution time. Each message sent incurs overhead (e.g. processing headers, extracting messages from the network, and payload), and this research investigates methods to reduce the cost of exchanging messages and reduce redundant information. The aim of this research is to test the hypothesis that clustering messages before sending them can reduce the overall communication cost and improve performance. A clustering mechanism can improve performance by aggregating multiple messages into a single, larger message. We investigate factors that impact the effectiveness and performance of message clustering to speed up the performance of a distributed ABS and provide a quantitative analysis. We also propose a new adaptive scheme that dynamically determines when to utilize message aggregation and determine the number of messages clustered. We compare this with non-adaptive clustering and

an unclustered approach.

Our empirical study and adaptive mechanism are implemented in the Scalable Agent-based Simulation SYstem (SASSY), a Java based distributed simulation kernel developed by the Distributed Simulation Lab at the University of Georgia [1]. SASSY is aimed to leverage advances in the field of parallel discrete event simulation (PDES) for agent-based simulations. It uses an optimistic synchronization protocol and was designed and developed to overcome the scalability issues that exist in current agent-based frameworks. Our implementation focuses on the Interest Management Logical Process (IMLP) mechanism implemented in SASSY [2]. Each agent in the simulation subscribes to a particular IMLP using a publish / subscribe protocol. The agents transmit messages and publish status updates via IMLPs. In our approach, messages to agents on remote machines are clustered at a local IMLP, forwarded to a remote IMLP, and then unclustered and sent to the receiving agent (see Figure 2).

The remainder of this paper is organized as follows: in the next section, we review related work from the artificial intelligence and simulation communities. In section III we present our implementation in detail. In section IV we present our results, and finally we conclude the paper with a discussion of future work.

II. Related Work

There has been some work done in the area to reduce communication costs in both the networking and simulation community. Our work is inspired by this previous work, but differs mainly in that it leverages knowledge available from the application layer and that it is adaptive in both the number of messages clustered and that it avoids clustering when it is determined advantageous. Our work is also unique in that it runs in a distributed agent-based simulation paradigm driven by a discrete-event simulation executive.

In the networking research community, Roy Friedman and Robbert van Renesse [3] compared the throughput and latency of two protocols (Tomfc and Dysfc) with and without message clustering. Their technique involved buffering the application messages for a short period of time and then sending them as a single packed message. Their study showed that message clustering improved both latency and throughput by two orders of magnitude. This improved performance was attributed to the fact that packing reduces the total number of bytes sent by replacing multiple packet headers with a single header for the clustered packet. This also causes less contention for network hardware, and fewer interrupts to handle messages.

Message aggregation techniques have also been proposed for sensor networks [4]. With the Application Independent Data Aggregation (AIDA) approach, He et al. made decisions based on the lower network layers instead of the application layer. Their approach has shown promising results for simulation. Other adaptive message aggregation protocols for sensor networks include RAP [5] and SPEED [6]. These protocols use the neighborhood (node) information like network congestion and traffic to make an informed decision about message routing and aggregation. SPIN [7] makes adaptive decisions to participate in data aggregation based on the cost of communication.

Other related techniques focus on aggregating data instead of messages. One of these techniques, Center at the Nearest Source (CNS) aggregates data at the source nearest to the destination [8]. In Shortest Path Trees (SPT), data is propagated along the shortest path from source to the destination and aggregated at common intermediate hops along the way.

Clustering of messages for agent-based systems in PDES is a relatively new concept. Takahashi and Mizuta addressed the message transmission costs in a large and complex agent-based simulation system on a large multi-node super computer, BlueGene, by clustering heavily communicating agents on the same node so that these agents can communicate via shared memory instead of remote message passing [9]. They essentially used a load-balancing approach to reduce communication costs. Their research shows promising performance results for 2 and 4 remote nodes connected via gigabit Ethernet. Our approach differs by using message clustering instead of load balancing to reduce communication costs. We leverage shared memory to communicate with agents that reside on the same node, and plan to incorporate load balancing in the future to further improve performance. A similar protocol in the networking community that promotes local communication is LEACH [10], a high-layer protocol that provides clustering and local processing to aggregate sensor data to reduce global communication.

Agents in our application use a publish and subscribe protocol between agents and Interest Managers (IMLPs). Each IMLP manages a subset of agents depending on where the agents reside in the environment [2]. Lees et al proposed a related approach where the world state was maintained by a tree of special logical processes known as Communication LPs (CLPs). The state of the environment is shared amongst these CLPs and any reads and writes to the environment state are facilitated by these processes. CLPs perform load balancing by swapping state variables when it is advantageous to reduce the total cost of access [11]. These researchers later studied rollback reduction by analyzing access patterns to CLPs. Since only certain accesses (e.g. premature reads) actually need to be rolled back, altering the Time Warp algorithm to take this into account reduces the total number of rollbacks (and thus the computation time) in a Multi Agent Simulation [12]. We intend to study these methods to reduce rollbacks in SASSY and further improve the performance of clustering.

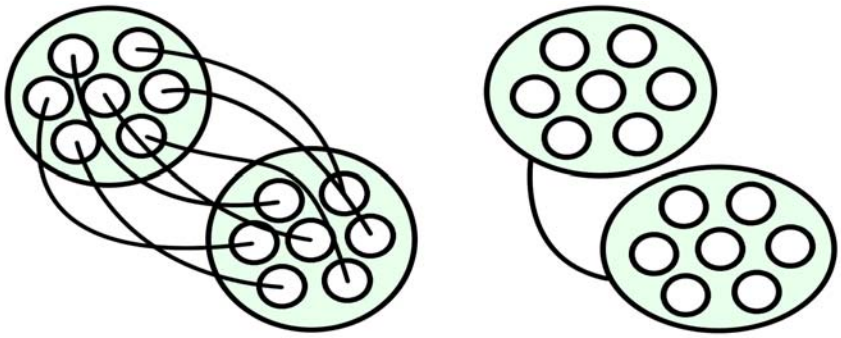

Fig. 2. Left: Agent Communication Across Grid Zones - Right: IMLP Message Clustering Approach

- Subscribe
- Unsubscribe
- Enter
- Leave
- Update
- **Ping**
- **Cluster**
- **Uncluster**

Fig. 3. SASSY Message Types - New Types are in Bold

III. Approach

To evaluate the effect of various message-clustering parameters on performance, we implement our own message-clustering scheme in SASSY and evaluate the scheme experimentally. In SASSY, agents are represented by a logical process (LP). These LPs maintain the most updated version of the environment visible to the agent. The SASSY middleware decomposes the sample space into a grid of cells to improve efficiency and scalability. An Interest Management Logical Process (IMLP) manages a cell or set of cells in the gridded environment. The IMLP implements a publish/subscribe system for LPs to ensure that all agents have a consistent view of their environment, unlike other agent-based systems that use a centralized global view of the environment for all agents [13], [14]. Every agent is mapped to at least one IMLP. We leverage this concept to implement our message-clustering algorithm, which is based on adding a few new message types into the existing SASSY system.

SASSY agents currently send five types of messages: subscribe, unsubscribe, enter, leave, and update. **Subscribe** messages allow LPs to monitor all messages that are published to a given IMLP. Similarly, the agent can send an **unsubscribe** message to stop receiving updates from a given IMLP. An **enter** message notifies the IMLP that an agent will be modifying the grid region that the IMLP manages. This may be something as simple as the agent moving through the region. Like an unsubscribe message, **leave** messages are sent by agents to inform an IMLP to stop listening for the given agent's updates. Once an agent enters an IMLPs grid space, it will send update messages that need to be relayed by the IMLP to all subscribing agents. To implement our message clustering mechanism, we added three new message types to SASSY (see Figure 3). **Ping** messages are sent amongst agents. The current IMLP that an agent is publishing information to will receive these **ping** messages and potentially cluster them. Clustered messages are created using a hash map to associate the destination LP with the message. This helps IMLPs route unclustered messages to the proper destination agents. Messages are clustered based on the destination agent's IMLP and then transmitted as one large **cluster** message. Once an IMLP receives a **cluster** message, it will uncluster the large message and forward the individual messages to the agents subscribed to it using an **uncluster** message. Agents will typically reside on the same machine as their IMLP, so this transfer can usually be done using shared memory instead of remote messaging. In previous work, Vulov, He and Hybinette showed that relaying messages through IMLPs showed better performance than sending messages directly between agents [15].

In this work we assume that agents generally multicast messages to a subset of agents in their environment instead of broadcasting to all agents. This is also true in biological systems such as ants which only communicate within their sensor range. The agents located in a cell at a particular instant will send messages only to agents in nearby cells. This ensures the message transmission is limited to multicast. To evaluate whether clustering messages in this environment is advantageous, we implemented two algorithms: fixed clustering and adaptive clustering. We compared these algorithms against a non-clustering approach. In the following subsections we will discuss these approaches.

A. Fixed Clustering

In the fixed clustering scheme a fixed number of messages (of variable payload size) are combined and then sent. There is also a configurable delay parameter that will cluster any buffered messages after a given period of time even if the cluster amount threshold has not been reached. This ensures

that all messages are delivered in a reasonable amount of time. This algorithm always clusters without taking parameters such as message size and network traffic into account. This behavior is depicted in Figure 4.

We observed that under some scenarios, clustering added a significant overhead. This overhead is addressed by our adaptive clustering mechanism that avoids clustering when it is not advantageous.

B. Adaptive Clustering

To avoid clustering overhead when advantageous, we propose an algorithm that adapts to the network traffic within the SASSY kernel. This is similar in spirit to the approach taken in AIDA [4] for sensor networks. Here the aim is to add an adaptive behavior that clusters when advantageous and avoids clustering otherwise.

To implement adaptive clustering we implemented message queues between the application layer and the SASSY kernel. All messages sent to the IMLP to be routed to respective agents must pass through this message queue. Our adaptive scheme monitors this message queue to ensure that it contains at least one message waiting to be processed by the kernel. If so, it infers that the network traffic is high and performs clustering. However, in scenarios where we observe an empty message queue, the kernel will transmit messages in their original form without aggregation. This approach gave us an adaptive clustering mechanism based on internal network traffic. When this approach was compared with fixed clustering we found that for equal threshold and end-to-end delay values the adaptive approach performed much better than the fixed one.

IV. Results

We performed four experiments to show the benefits of clustering in an agent-based simulation system. We first determined that sending fewer, larger messages is more efficient than lots of small messages. Based on these results, we developed a clustering approach and tested it against the unclustered approach by altering the number of agents in the simulation and the message payload sizes. In our final experiment, we tested the performance of the adaptive clustering mechanism over fixed clustering. All experiments were conducted in a distributed environment with 10 IMLPs, 10 PEs, 50 - 500 agents and message payload sizes ranging from 1 - 8000 bytes. Tests were executed on dual-core 2.6 GHz workstations with 4GB of RAM and networked with gigabit Ethernet. The results of our experiments are detailed in the following subsections.

A. Message Clustering

Our first experiment was conducted to assess the impact on communication overhead resulting from sending large packets as opposed to several smaller packets. This experiment was conducted using 10 IMLPs spread across 10 PEs with 500 agents. To understand the message clustering impact we selected values that depend on payload size. The size of message payloads was fixed, and the experiment was repeated four times sending a variable number of fixed-size messages each run. In these runs, the total payload was fixed and split into multiple smaller messages. The results are shown in Figure 5. The experiment was repeated with four different message sizes from 1KB to 8KB. As expected, for each message size it was much faster to send a single packed message instead of multiple smaller messages. This demonstrated that clustering could potentially improve run times in agent-based simulations.

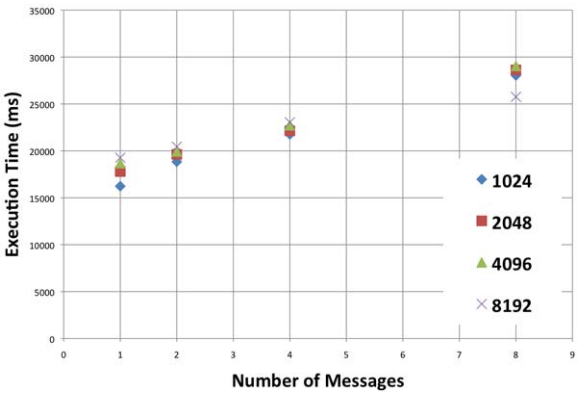

Fig. 5. For each size, a single message ran faster than multiple smaller messages (lower is better)

B. Varying Numbers of Agents

In our next experiment, we began to explore a clustered approach. Experiments were run on 10 machines (one PE per machine). We varied the number of agents in the system and each agent sent 50 messages that were randomly sized between 1 and 1000 bytes. After sending each message, agents would wait a random amount of time up to 10ms to simulate computation. Quantitative results for this experiment are shown in Figure 6. The X-axis shows the number of agents and the Y-axis shows the execution time in milliseconds. Each data point represents the average execution time over five runs. The clustered approach was initially faster than the unclustered approach, but as the number of agents in the system increase, the two approaches converge. We ran an additional trial with 1000 agents and found that the unclustered approach performed better. We believed this to be a result of the increased rollbacks caused by too many agents assigned to each Interest Manager. We repeated this trial with 100 IMLPs instead of 10. With a lower agent-to-IMLP ratio, the clustered approach performed with a speedup of 1.8 over the unclustered one.

C. Varying Message Size

The next experiment was to highlight scenarios when the message clustering approach may not be advantageous and to

```
private void PING(Message msg)
{
    numClustered++;
    destination = imlpFor(msg.getAPPid());

    if(buffer.containsKey(destination))  {
        buffer.get(destination).add(msg.getMyMessage());
    }
    else  {
        buffer.put(destination, addMessage(msg.getMyMessage()));
    }

    if(numClustered > threshold || currentLocalTime > maxClusterTime)  {
        for(String destination : buffer.getKeys())  {
            sendMessage(buffer.get(destination), destination);
        }
        resetBufferAndClusterTime();
    }
}
```

Fig. 4. Fixed Clustering Approach

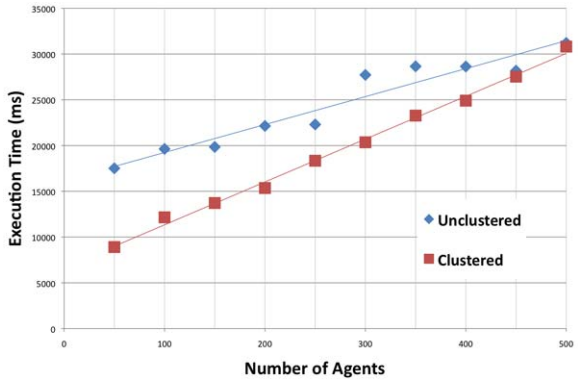

Fig. 6. Runtime vs. Number of Agents on 10 IMLPs (lower is better)

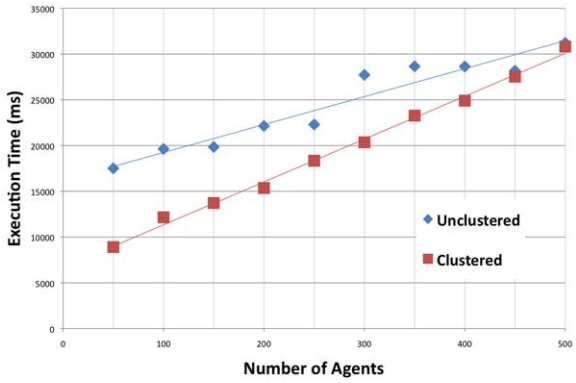

Fig. 7. Runtime vs. Message Size (lower is better)

give us insight into how to parameterize adaptive clustering. To study this we ran multiple iterations of fixed clustering with message sizes between 20 and 500 bytes. We measured many small incremental changes in message size to observe the impact of message clustering with smaller messages and note the threshold value at which clustering starts to impact the system performance. This experiment was conducted on 10 PEs, 10 IMLPs and 500 agents. The results of this experiment are shown in Figure 7. We observed that message size does not have a significant impact on execution time in either method. As you can see, each method takes approximately the same amount of time independent of message size. This shows that the number of messages sent, and the number of agents in the system have a greater impact on execution time than message size.

D. Adaptive vs. Fixed Clustering

Our final experiment was conducted to test our adaptive clustering approach. There are a number of factors that impact the performance of message transmission including message size and processing time, latency, bandwidth and external work loads. To test the adaptive clustering approach we focused on number of agents. We varied payload from 1 - 1000 bytes, and number of agents from 50 - 500. The experiment was distributed over 10 PEs with 10 IMLPs. Once again the results in Figure 8 represent the average of 5 runs. Adaptive clustering showed much more speedup than fixed clustering. The adaptive clustering approach takes into account the internal traffic that depends on the processing time of the SASSY kernel and also the time taken to copy messages between network and local storage. Clustering is performed when the queue that

holds unprocessed messages contains unsent messages. Hence we observe that for the same amount of delay, performing clustering of messages was not always helpful compared to the adaptive approach.

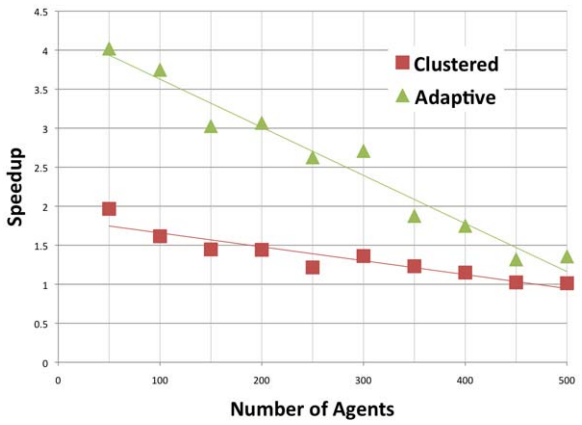

Fig. 8. Speedup of Adaptive and Fixed Clustering vs Unclustered Approach (higher is better)

V. Conclusions and Future Work

In this work we have shown that message clustering, or "piggybacking," can improve performance in large scale distributed agent-based simulation. We demonstrated the benefit of message clustering in a PDES system. We evaluated traditional messaging as well as fixed and adaptive clustering approaches by implementing them in SASSY, a Java agent-based PDES system. Our results indicate that the decision to cluster should depend on message payload size and number of agents in the system. Fixed clustering may not always be beneficial, like when the agent LP to IMLP ratio is high, and the overhead of clustering and unclustering messages dominates. The adaptive clustering approach monitors the system to determine when clustering will be beneficial. This approach always outperforms both the unclustered approach and fixed clustering approaches.

This research restricts agents to a single processing element (PE). To achieve a more realistic agent-based scenario, we would like the agents to be able to migrate across PE nodes dynamically. This would allow us to investigate communication patterns for complex, realistic agent-based systems such as colonies of ants or bees. Movement across nodes would also help us distribute the load of the simulation system by aggregating highly interactive agents onto a single node so that they could communicate via shared memory instead of sockets. This would help us investigate the effect of external factors like load balancing on our adaptive clustering technique.

Our clustering approach was always tested with a threshold of one time step. If an Interest Manager received any messages that were at least one time step ahead of local time, the clustered messages would immediately be sent and new clusters started. In the future we will explore the effect of longer threshold times on rollbacks and execution time.

References

[1] M. Hybinette, E. Kraemer, Y. Xiong, G. Matthews, and J. Ahmed, "SASSY: A design for a scalable agent-based simulation system using a distributed discrete event infrastructure," in *Proceedings of the 2006 Winter Simulation Conference*, Dec. 2006, pp. 926–933.

[2] T. He and M. Hybinette, "A comparison of interest manager mechanisms for agent-based simulations using a time warp executive," in *pads-08*. IEEE Computer Society, 2008, pp. 157–162. [Online]. Available: http://doi.acm.org/10.1145/1381309.1382304

[3] R. Friedman and R. van Renesse, "Packing messages as a tool for boosting the performance of total ordering protocols," in *Proceedings 6th International Symposium on High Performance Distributed Computing (HPDC '97) (6th HPDC'97)*. Portland, OR, USA: IEEE Computer Society, Aug. 1997, pp. 233–242.

[4] T. He, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "AIDA: Adaptive application-independent data aggregation in wireless sensor networks," *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 2, pp. 426–457, May 2004.

[5] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, "RAP: A real-time communication architecture for large-scale wireless sensor networks," in *IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE Computer Society, 2002, pp. 55–66. [Online]. Available: http://computer.org/proceedings/rtas/1739/17390055abs.htm

[6] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher, "SPEED: A stateless protocol for real-time communication in sensor networks," in *International Conference on Distributed Computing Systems (ICDCS-2003)*. IEEE Computer Society, 2003, p. 46.

[7] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. ACM, 1999, pp. 174–185.

[8] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *Proceedings of the 22nd International Conference on Distributed Computing Systems*. IEEE, 2002, pp. 457–458.

[9] T. Takahashi and H. Mizuta, "Efficient agent-based simulation framework for multi-node supercomputers," in *Winter Simulation Conference*, L. F. Perrone, B. Lawson, J. Liu, and F. P. Wieland, Eds. WSC, 2006, pp. 919–925. [Online]. Available: http://doi.acm.org/10.1145/1218112.1218281

[10] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS '00)*, 2000. [Online]. Available: http://computer.org/proceedings/hicss/0493/04938/04938020abs.htm

[11] M. Lees, B. Logan, R. Minson, T. Oguara, and G. K. Theodoropoulos, "Distributed simulation of MAS," in *Multi-Agent and Multi-Agent-Based Simulation, Joint Workshop (MABS 2004)*, ser. Lecture Notes in Computer Science, P. Davidsson, B. Logan, and K. Takadama, Eds., vol. 3415. Springer, 2004, pp. 25–36.

[12] M. Lees, B. Logan, and G. K. Theodoropoulos, "Using access patterns to analyze the performance of optimistic synchronization algorithms in simulations of MAS," *Simulation*, vol. 84, no. 10-11, pp. 481–492, 2008. [Online]. Available: http://dx.doi.org/10.1177/0037549708096691

[13] P. F. Riley and G. F. Riley, "SPADES – a distributed agent simulation environment with software-in-the-loop execution," in *Proceedings of the 2003 Winter Simulation Conference (WSC-2003)*, Dec. 2003, pp. 817–825.

[14] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the International Conference on Advanced Robotics*, Coimbra, Portugal, Jul 2003, pp. 317–323.

[15] G. Vulov, T. He, and M. Hybinette, "Quantitative assessment of an agent-based simulation on a time warp executive," in *Proceedings of the 2008 Winter Simulation Conference(WSC-2008)*, S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler, Eds. WSC, 2008, pp. 1068–1076. [Online]. Available: http://dx.doi.org/10.1109/WSC.2008.4736175