# A SYNCHRONIZATION PROTOCOL FOR DISTRIBUTED AGENT-BASED SIMULATIONS WITH CONSTRAINED OPTIMISM

Dirk Pawlaszczyk
Steffen Strassburger

School of Economic Sciences
Ilmenau University of Technology
Helmholtzplatz 3, 98693 Ilmenau, GERMANY
Email: pawlaszczyk@hotmail.com / steffen.strassburger@tu-ilmenau.de

**KEYWORDS**
Agent-based simulation, optimistic synchronization, contrained optimism, FIPA interaction procotols.

**ABSTRACT**

Agent-based simulation (ABS) is a paradigm which has received much attention within the last years. For enabling industrial use of ABS scalable solutions are required which can be executed on a distributed computing architecture. Such solutions must be capable of simulating complex models with hundreds or more complex deliberative agents with really autonomous behavior.
In this article we argue that only optimistic synchronization protocols are potentially capable of providing the required performance. We suggest a synchronization protocol with constrained optimism which exploits specific characteristics of communication patterns within agent based simulations. Furthermore the presented protocol includes appropriate methods for GVT computation and fossil collection in distributed ABS as well as mechanisms to ensure repeatability.

## INTRODUCTION

In agent-based simulation (ABS) real world systems are modeled using multiple agents. The modeled system emerges by interaction of the individual agents as well as their collective behavior. Agents typically send messages with respect to some communication protocol. In this context a software agent is defined as a program that acts autonomously, communicates with other agents, is goal-oriented (pro-active) and uses explicit knowledge.
Agent-based modeling and therefore agent-based simulation seems to be an appropriate tool for domains characterized by discrete decisions and distributed local decision makers. With growing complexity of agent based models, the scalability of a simulation environment becomes a crucial measure. To simulate an increasing number of entities, the underlying simulation system needs to be scalable, thus creating an immediate demand for distributed simulation.
Although ABS has received a lot of attention in recent years there are rather few contributions in place that deal with the problem of scalable distributed agent-based simulation. The objective of the research presented here is to discuss in detail an approach for a scalable time synchronization algorithm which takes advantage of the specifics of the agent-based simulation approach effectively.

## BASICS OF AGENT-BASED SIMULATION

### Agent Technology and Standards

When using agent-based design approaches it is often the interplay between multiple agents, which one is interested in. This leads to the term of multi-agent systems (MAS). An MAS is generally considered a system composed of multiple autonomous agents which can interact with each other. Multi-agent systems can be used to solve or describe problems which are difficult or impossible to solve/describe with individual agents or a monolithic system.
It is often assumed that agents in MAS are executed and interact with each other in real-time, i.e., there is typically no separate logical time representation within an agent as it is known from paradigms like discrete event simulation. Please note that this is often different when talking about ABS, as discussed further down in the paper.
Agents in MAS are only capable of exchanging knowledge and interacting when they use a common language understood by all agents. The Foundation for Intelligent Physical Agents (FIPA) is an organisation founded with the objective of creating a framework architecture for the interaction of heterogeneous agent systems. A central point of this effort are standards for message-based communication of agents and multi-agent systems. The structure of a message is defined by the Agent Communication Language (ACL) (FIPA 2002). Within messages the communicative act, i.e. the intention of the sender, is identified using performatives like “inform”, “request”, “agree”, etc. Figure 1 gives an ACL message example which is part of an auction protocol.

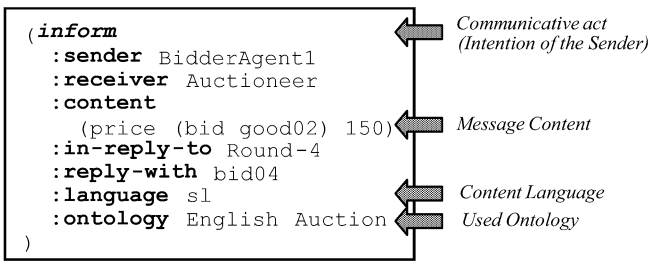


**Figure 1: Example for an ACL-Message**

Most importantly for the further discussions, FIPA also defines specifications for interaction protocols (FIPA 2002a). These interaction protocols define typical sequences of messages or patters, how agents may interact. The resulting dialogs between agents always follow this same pattern. A simple example of this is shown in Figure 2.

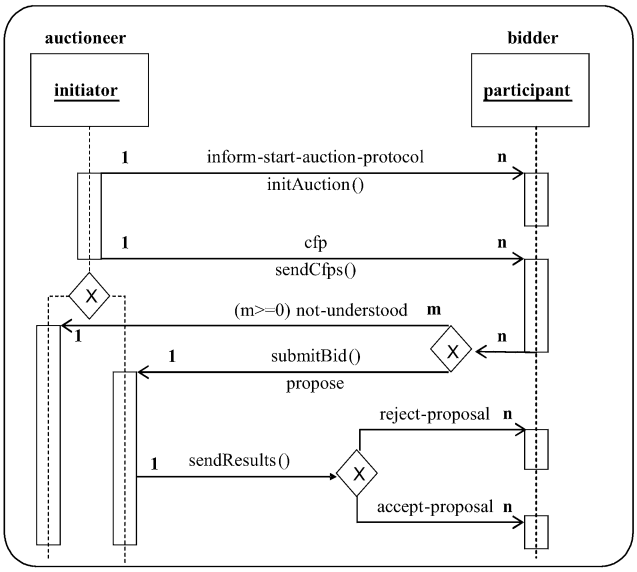


**Figure 2: Example for an auction protocol**

The figure depicts the possible lines of communication between an auctioneer agent and multiple bidder agents. The auctioneer first has to request bids from the participating bidders, before he can accept a proposal and inform the successful bidder.
The possible types of the messages exchanged within this interaction protocol and their sequence is independent from the actual message content. It is, for instance, completely irrelevant whether the auction concerns a pencil or a car.
The most frequently used interaction protocols between agents are firmly defined within the described FIPA standards (compare Table 1).

**Table 1: Some FIPA Interaction Protocols (FIPA 2002a)**

| Title | Description |
|---|---|
| FIPA Request Interaction Protocol Specification | Allows one agent to request another to perform some action. |
| FIPA Query Interaction Protocol Specification | Allows one agent to request to perform some kind of action on another agent. |
| FIPA Request When Interaction Protocol Specification | Allows an agent to request that the receiver perform some action when a given precondition becomes true. |
| FIPA Contract Net Interaction Protocol Specification | Allows one agent (the Initiator) to take the role of a manager which wishes to have some task performed by one or more other agents (the Participants) and further wishes to optimize a function (e.g., price) that characterizes the task. For a given task, any number of the Participants may respond with a proposal; the rest must refuse. Negotiations then continue with the proposer. |
| FIPA Propose Interaction Protocol Specification | Allows an agent to propose to receiving agents that the initiator will do the actions described in the propose communicative act when the receiving agent accepts the proposal. |

The interaction protocol to which a message belongs is contained within a message field. By using a certain protocol the agent commits to react to requests of the protocol in the predefined format.
The FIPA Interaction Protocols are designed for usage in MAS in general. In our work, we adopt them to be used in ABS and we take advantage of the fact that they define commonly used interaction *sequences*.

### Agent-based Simulation (ABS)

The term agent-based simulation (ABS) describes the modeling and simulation of real systems with the help of agents, which interact within a simulation model.
The usage of agent based approaches for modeling and simulation promises greater flexibility and better abstraction capabilities when describing the behavior of systems with many active (or "intelligent") components. In agent-based simulations agents can represent workers, machines, carriers, etc. which can all have their autonomous behaviour.
There is a wide variety of development environments for agent-based simulations, mostly from academic sources. Current examples include Cougar, Farm, James II, Repast, Samas, Sassy, and many more.
Please note that most of these environments use event based mechanisms for advancing logical simulation time. This shows the strong influence that paradigms like discrete event simulation have had on ABS. In fact, one could argue that ABS as a modeling philosophy is quite similar to modeling with object-oriented simulation tools with process-oriented world views like SLX (Henriksen 1997).
In the further discussion we limit the scope of our work to this type of ABS. We do not consider ABS which operate in a real-time manner like common MAS.

### Distributed Simulation and ABS

According to Fujimoto (2000) distributed simulation (DS) is a technology that enables a simulation program to be executed on distributed computer systems.
Agent-based modeling and simulation environments do not necessarily have to be build in a distributed fashion (Uhrmacher and Gugler 2000). However, when scalability issues have to be considered, the parallel or distributed execution of agents is the only technology offering hope for increased performance when the problem size (i.e. the number of agents and their complexity) increases beyond that what a single machine can simulate or process. Main motivation for suggesting the usage of distributed simulation for agent-based models is therefore the scalability aspect.
For enabling scalable agent based simulation, both hardware and software (i.e. the applied algorithms) has to be scalable. The focus of the discussion in this paper is on the software aspect, specifically on a synchronization algorithm that scales well for agent based models and their specific characteristics.

## OPTIMISTIC SYNCHRONISATION OF DISTRIBUTED AGENT-BASED SIMULATIONS

Synchronization protocols as the core technology needed for distributed simulation can be classified into the two main

categories of conservative and optimistic protocols (Fujimoto 2000).
Conservative protocols implement mechanisms that prevent a member of a distributed simulation from processing messages out of time stamp order, thus maintaining strict causality. Conservative synchronization protocols are rather easy to use and implement, but their performance depends highly on a value called Lookahead. Lookahead is a guarantee from a simulation that it will not generate any messages with a time stamp smaller than its current time plus the value of Lookahead. If a simulations' current time is T, and its Lookahead is L, any message generated by the federate must have a time stamp of at least T+L.
Lookahead is hard to extract and always depends on the application context. For agent-based simulations, the situation comes close to the worst case scenario, as their interaction protocols often require immediate answers resulting in a Lookahead requirement. In this case, conservative synchronization protocols almost completely inhibit parallelism within the distributed simulation.
Optimistic protocols do not impose the requirement to process events in strict time stamp order. Simulations using optimistic synchronization can process received messages although there maybe future messages with a smaller time stamp. To maintain causality, these approaches detect and recover from causality errors, which may be introduced by processing events before it is safe to proceed. The major advantage of these approaches is that they allow the exploitation of parallelism in situations where it is possible that causality errors might occur, but in fact they do not occur. The Time Warp protocol is an example of an optimistic synchronization mechanism.

**Experimentation Framework**

Optimistic protocols are more complex to implement than conservative protocols, but the following statement of Fujimoto (2000) certainly holds much truth: "If one's goal is to develop a general purpose simulation executive that provides robust performance across a wide range of models […] optimistic synchronization offers greater hope." Consequently, our research has focused on this approach and has tried to intelligently combine it with ABS in order to eliminate some of the problems which are inherent to optimistic synchronization techniques.

We have developed a simulation kernel to enable efficient simulation of large-scale agent based models. Therefore, we have added parallel discrete event simulation (PDES) functionalities on top of an existing agent middleware to get support for optimistic simulation.

Our implementation is based on the Java Agent Development Environment (JADE), a generic framework for development of agent based applications (JADE 09). JADE offers an appropriate middleware to simplify the implementation of multi agent systems. It is widely used in academia. Since JADE is compliant to the FIPA standard, a high degree of interoperability is guaranteed. Moreover, the JADE messaging sub system scales well, even for heaviest message traffic (Vitaglione et. al 2002). JADE provides many built in features like remote method invocation (RMI), serialization, and agent management tools that allow a distributed model to be easily developed. Because the program is written in the Java programming language the implementation is portable across a wide range of platforms.

Applying conventional PDES models and techniques to MAS is more complicated than one might think, since we have to regard specifics of agent technology like a particularly high communication demand and dynamic topologies, without degrading performance. At the heart of the simulation executive there is a new optimistic synchronization algorithm. Furthermore we have implemented an efficient decentralized scalable algorithm for computation of GVT (global virtual time) taking into account transient messages without using blocking barriers. Other basic features of the simulator include automatic state saving, repeatability of simulation runs using a tie-breaking algorithm (Mehl 1992) and simulation support for FIPA compliant agent models. All PDES-functionalities were added to the JADE middleware as platform services using the standard plug-in concept of this agent-framework. This enabled us to test the feasibility of the newly developed algorithms.
Programming simulation models with the simulator is straightforward. Individual agents are programmed by the application developer using the standard agent-based sense-think-act paradigm. Agent processes are autonomous in the sense that they hold and manage their own events, and are optimistic in their event processing. State saving and synchronization of agent processes is done in background.

**Optimistic Processing**

A scalable architecture by default requires it to be composed of multiple processors/computers, i.e., so that the number of used resources can grow as the problem size increases. Searching for a synchronization technique we have disqualified conservative approaches because of the inherent zero lookahead requirements in many agent interaction protocols. Hence, optimistic synchronization seems to be the most promising approach, because it (at least theoretically) provides enough performance for a wide range of models.
However, within pure Time Warp implementations, a common problem can be caused by rollback cascades as there are no constraints on the relative distance between logical processes (LPs). One can call this a problem of "too much optimism". Every LP processes events almost independently of the progress of other LPs' timelines. Consequently, the probability of incorrect computations (i.e. causality errors) is very high. If the time to perform a rollback is high, i.e. many states have to be rolled back, the performance of the simulation rapidly decreases.

A good time management algorithms therefore has to avoid situations like these while still ensuring a high degree of parallel execution.

The basic idea of our suggested approach is very straight-forward: We suggest to limit the level of optimism in the applied time warp protocol by using extra-knowledge extracted from the used agent interaction protocols. Communication following these interaction protocols is one of the key features in agent technology.

Messages are sent out from a sender to one or more receiver(s). Messages are encoded in an *Agent Communication Language (ACL)*, an external language that defines the intended meaning of a message by using performatives. A series of messages produces a dialog. A

dialog normally follows a predefined structure – the *Interaction Protocol* (IP). The FIPA *Request Interaction Protocol* for example allows an agent to request another agent to perform some action. The responding side needs to decide whether to accept or refuse the request. In any case, the message receiver has to respond. Even if the receiver cannot interpret a message, the specification prescribes to send a least a not-understood message.

We exploit this characteristic: the communication almost always follows a known sequence of messages. In normal Time Warp every new event is immediately sent over the network to its corresponding receiver where it can be executed immediately. Our approach leverages delays on the event execution based on information about the current state of the interaction protocol. Instead of immediately processing every incoming event message the event is delayed using an adaptive rule:

**Definition 1** (wait for rule): Given agent $a_1$ which has sent a message $m_1$ to agent $a_2$, and assuming that there is at least one valid required reply $m_2$ for $m_1$, the rule "wait for" is defined as follows: If the expected reply message was not received yet, the agent $a_1$ must wait for this particular message, before going on to process the next message.

This rule basically requires agents to wait for a reply if a communication which is part of an interaction protocol has been started. In the *FIPA-request-protocol* for example, if an agent has agreed to do something for its opponent he automatically commits himself to send an *inform-done-message* as soon as he has finished the task. If he did not succeed he has to send a *refuse-message*. In any case the agent always has to reply. Accordingly, for every message $m_i$ which is received while agent a is waiting for message $m_k$ from a sender different to the sender of $m_i$, this message is buffered. The execution of $m_i$ is delayed.

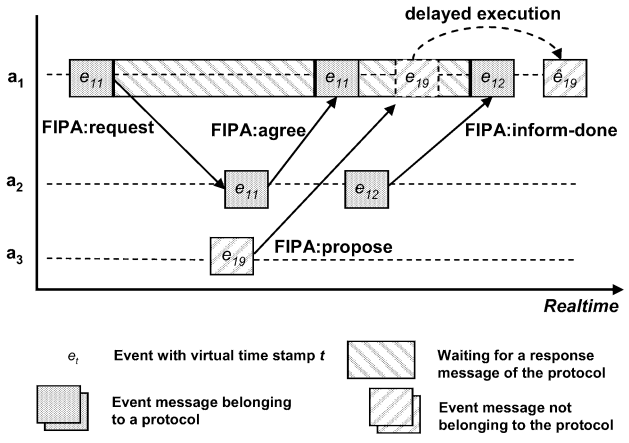


**Figure 3: Delayed event execution based on protocol information. Agent $a_1$ receives a proposal from Agent $a_3$, while he is waiting for an inform-done message of Agent $a_2$. Instead of immediately processing the incoming message, the execution is delayed. Thus, event order is preserved and still valid**

In Figure 3 an example is given to demonstrate the effect of delayed execution. The events in this example have distinguishable time stamps only for reasons of clarity. The suggested synchronization protocol can handle simultaneous events just in the same way.

In the example, the immediate execution of event $e_{19}$ from $a_3$ normally would cause the agent process $a_1$ to rollback when message $e_{12}$ is received at some later point in time. With the new policy in place this situation can easily be avoided. Instead of immediately processing message $e_{19}$, agent $a_1$ has to wait for the reply message from agent $a_2$. This is because there is an external knowledge that the active interaction protocol will sooner or later require $a_2$ to send a message "FIPA: inform-done" (expressed as $e_{12}$). The wait-for-rule basically formalizes this behaviour. Therefore event $e_{19}$ has to be buffered thus preserving the relative event order. This approach minimizes the potential for incorrect execution of events as far as knowledge from the interaction protocols can be extracted.

The process of waiting for a certain message could be interpreted as a conflict to the asynchronous nature of agent execution, but in fact, it is not a contradiction, rather, it is a natural approach common in PDES to maintain causality. Agents in our simulations must behave consistently with the interaction protocol specification. If agent $a_1$ in the example above executed $e_{19}$ immediately upon reception, a rollback would be unavoidable when $e_{12}$ is received. Since the agent has the external knowledge that $e_{12}$ will in fact occur, it is straight-forward to avoid this situation. It should also be noted, that the result of the simulation is not affected in any way by this approach.

Further to the basic example given above, the implementation of the presented approach has to take into account some special cases where exceptions to this rule may be needed. First, let us have a look at the general decision tree which is passed whenever a new message is sent from one agent to another (Figure 4).

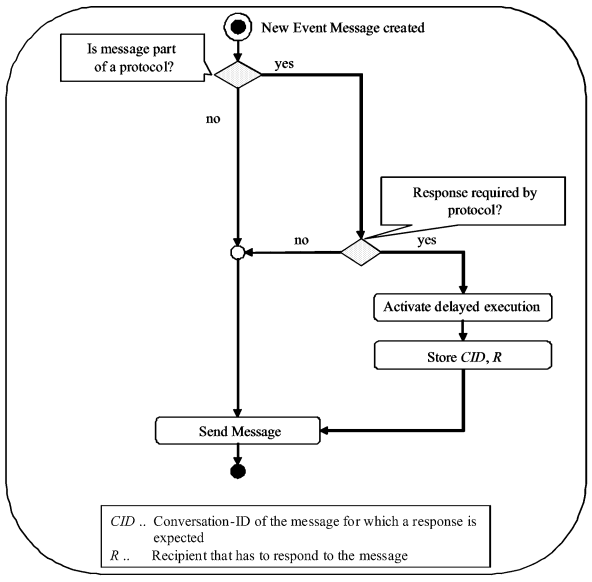


**Figure 4: Algorithm for sending new messages and enabling constrained optimism**

If the message is part of an interaction protocol and a response is required by this protocol, the delayed execution is activated and the message is flagged. All further received messages are subject to the wait for rule and their execution is potentially delayed.

However, there may be some situations where exceptions to the wait-for rule may be required. In certain situations an agent must exceptionally be allowed to execute events, even if it is waiting for a reply message. Consider, for example, the following potential *deadlock* situations. Assuming an agent $a_1$ waits for agent $a_2$, and at the same time agent $a_2$ waits for agent $a_1$. This may be the case since both independently have sent a message to each other, each being part of a different conversation. Both agents would then become blocked as they are both waiting for a event message which will never occur. In this case an agent must be allowed to process a message from its opponent even if it is not the message content it was waiting for. Another exception are *cyclic dependencies*. Although not very likely, there may be situations when an agent receives a request within the same conversation, from a new communication partner different from its original opponent. This may be the case for example in multi-staged-protocols. In this situation, this new message has to be processed by the waiting agent before he can go to wait state again.
Figure 5 illustrates the decision tree at the receiver side which takes these special situations into account and bypasses the delayed execution if needed.

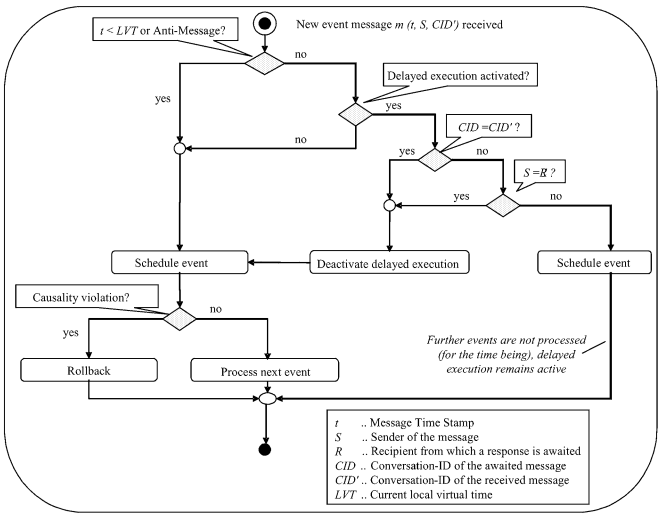


**Figure 5: Algorithm for processing received event messages taking into account necessary exceptions**

The so described synchronization algorithm is joining optimistic techniques with constrained optimism and can therefore be classified as a time warp with constraints. The proposed policy certainly cannot fully prevent rollback situations. However, it minimizes the risk for rollbacks caused by messages which causally belong to a common communication. For agent-based simulations this approach, according to our experiments, performs better than a pure Time Warp solution. Also, this approach is capable of maintaining an acceptable degree of parallelism required for scalable solutions.
The implementation effort of this solution is considerably low. The agent has to be provided with information about the structure of the used protocols at initialization time only. Depending on the protocol length, the policy is applied more frequently. Particular long interaction protocols, like the *fipa-contract-net* are most eligible.

## SUMMARY AND CONCLUSIONS

While agent-based simulation as a methodology has received much attention from the research community in the past, scalability and thus industrial applicability of this technology has been somewhat neglected. In this paper we have argued that only the efficient distributed simulation of agent-based models can provide the architectural basis for a scalable solution. We have further on argued that only the deployment of an optimistic synchronization protocol can yield the required performance. Further on, we have in detail presented a constrained optimistic synchronization protocol, which eliminates a significant amount of the drawbacks and risks that exist within the original optimistic time warp protocol.
This is achieved by taking advantage of specific characteristics that are inherent in many agent based interaction protocols making it therefore a good solution for scalable distributed simulation of agent based models which are based on logical simulation time. Empirical performance results for a variety of tests support this statement (Pawlaszczyk and Strassburger 2009).

## REFERENCES


FIPA. 2002. FIPA ACL Message Structure Specification. FIPA Spec 00061. Available via http://www.fipa.org/specs/fipa00061/SC00061G.pdf [accessed April 1, 2009].

FIPA. 2002a. FIPA Interaction Protocols. FIPA Specs 0026-0036. Available via http://www.fipa.org/repository/standardspecs.html [accessed April 1, 2009].

Fujimoto, R. 2000. *Parallel and Distributed Simulation Systems*. Wiley Interscience, 2000.

Henriksen, J.O. 1997. An introduction to SLX. In *Proceedings of the 1997 Winter Simulation Conference*, ed. Andradóttir, S., K. Healy, D. Withers, and B. Nelson, 559-566. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

JADE 2009. Homepage ot the JADE project: http://jade.tilab.com [accessed April 1, 2009].

Pawlaszczyk, D., S. Strassburger. 2009. Scalability in Distributed Simulations of Agent-Based Models. To appear in: *Proceedings of the 2009 Winter Simulation Conference,* eds. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. December 13-16, 2009. Austin, USA.

Mehl, H. 1992. A Deterministic Tie-Breaking Scheme for Sequential and Distributed Simulation. In: *Proceedings of 6th the Workshop on Parallel and Distributed Simulation*, ed. M. Abrams and P. F. Reynolds, 199-200.

Uhrmacher, A. M., K. Gugler. 2000. Distributed, parallel simulation of multiple, deliberative agents. In: *Proceedings of the 14th Workshop on Parallel and Distributed Simulation*, 101-108. Washington, DC, USA: IEEE Computer Society, 2000.

Vitaglione, G., F. Quarta, E. Cortese. 2002. Scalability and Performance of JADE Message Transport System. In: *Proceedings of the AAMAS Workshop on AgentCities*, Bologna, 2002.