Towards a Real-Time Agent Architecture – A Whitepaper

Lisa Cingiser DiPippo, Ethan Hodys,
The University of Rhode Island
Kingston, RI USA 02881
lastname@cs.uri.edu

Bhavani Thuraisingham
MITRE Corporation
Bedford, MA USA
thura@mitre.org;

Abstract

Applications such as military training simulations, and electronic commerce can benefit from the flexible and responsive nature of multi-agent systems. These applications have inherent timing constraints on the operations and interactions that the agents might perform. This paper presents a real-time agent architecture in which agents communicate, cooperate, coordinate and negotiate to meet the goals of a particular application under specified timing constraints. The architecture provides a real-time CORBA layer to handle underlying real-time communication. It also has a real-time agent communication layer in which agents interact via a real-time extension of a well-known agent communication language.

1 Introduction

Multi-agent systems provide a software environment in which autonomous agents coordinate, cooperate and negotiate to meet the goals of a particular application. Many real-time applications could benefit from the flexible and responsive nature of agents in a multi-agent system. For example, a military training simulation could incorporate agents to represent enemy forces. These "enemy agents" would react to the changing environment, and they would coordinate and negotiate to meet the goal of attacking or responding to an attack of the forces taking part in the simulation. While current multi-agent systems can provide the underlying software to build these autonomous agents, they do not have the capability to express timing constraints on actions, and to enforce these timing constraints. In the military training scenario, the simulation would have timing constraints on agent reactions to particular stimuli, such as movement of a tank or detection of an incoming missile. These timing constraints must be expressed in a formal way in the system, and there must be some mechanisms for enforcing them.

In this paper, we present a real-time agent architecture in which agents coordinate, cooperate and negotiate under specified timing constraints. The architecture includes a real-time agent communication layer in which a well-known agent communication language is extended to express timing constraints. It also includes a real-time CORBA layer to handle underlying communication among agents, and to enforce the expressed timing constraints.

The rest of this paper is organized as follows. Section 2 provides background on agents, multi-agent systems, existing agent architectures, and previous work on real-time agents. Section 3 describes the real-time agent architecture. Section 4 concludes and discusses future work in this area.

2 Background

In this section we provide definitions for some basic terms that we will use throughout the paper. We then go on to describe some of the recent work that has been done in the field of agent architectures and real-time agent research.

2.1 Definitions

The study and development of agents has been an active field of research for several years. The work involved in these studies has produced a wide range of uses for agents, and almost as many working definitions. However, in [1], a consensus definition is presented, and we use this for our definition of an agent. According to [1], an *agent* is a computer system, situated in some environment, that is capable of flexible autonomous action in order to meet its design objectives. This definition implies that an agent must be able to react to events that occur in its environment in order to maintain its goals. Further, along with being reactive, an agent must also be proactive. That is, it must be able to take initiative and be opportunistic when necessary in order to meet its stated goals. Finally, an agent must be social. It must be able to interact with other agents and with users to coordinate to meet common goals, and to negotiate to resolve conflicting goals.

As an example, consider an agent whose job is to retrieve data from a particular source for a user or another agent. This data retrieval agent knows when the user, or some other agent, requires certain data. It either retrieves the data itself if it has access to the required data, or it notifies another data retrieval agent that can get the data. The user does not need to request the data explicitly. Rather, the agent recognizes situations in which the data is useful or necessary to the user and it autonomously retrieves it.

When multiple agents work within the same system, we call such a system a *multi-agent system*. This type of system is in contrast to a computer system in which individual agents may exist, unaware of other agents in the system. Cooperating agents in a multi-agent system exist within a framework in which they can communicate, coordinate and negotiate to meet their goals. We call this framework the *agent architecture*.

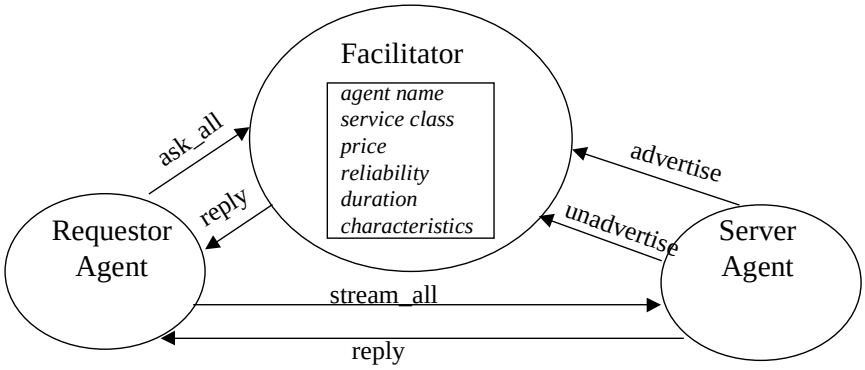

Figure 1 - Agent Facilitator / Matchmaker

Several models for agent communication have been developed: *agent to agent*, *agent broker*, and *agent matchmaker* [2]. In an agent to agent model, each agent knows the name of any other agents with which it might need to communicate. This is a completely distributed model in that there is no central coordinator among the agents. The latter two models both fall into a category of agent *facilitators*, in which a special agent is tasked with finding agents to fulfill services required by requestor agents. An agent broker is an agent that facilitates communication among other agents. In this model, each agent registers with a broker and advertises the services that it can perform on behalf of other agents. When an agent requests a service from the broker, the broker passes the request along to the agent that provides the requested service. If no such agent exists, that is, if the requested service has not been advertised to the broker by any agent, the broker responds to the requesting agent with a message. An agent matchmaker works in much the same way as an agent broker. However, when a request for service is made to a matchmaker, the matchmaker agent passes along a reference to the agent that can provide the service. That is, the matchmaker puts together, or matches, agents that can work together. Once this match is made, the agents can communicate with each other directly without the use of the matchmaker. Figure 1 illustrates the functionality of both a facilitator agent and a matchmaker agent.

Communication among agents and facilitators is typically achieved through an agent communication language. The Knowledge Query Manipulation Language (KQML) [3] is a widely used agent communication language that provides "performatives" to define the kind of interactions a KQML-speaking agent can have. These include performatives to request that an agent perform a task (`ask-one`), to provide other agents with certain information (`tell`), to watch another agent for a particular condition (`monitor`), and to register capabilities with a facilitator agent (`advertise`). Figure 2 displays some examples of how these performatives might be used in a multi-agent system for stock trading.

```
(ask-one
   :content (PRICE_IBM ?price)
   :receiver stock-server
   :language Prolog)
```

```
(advertise
   :language Prolog
   :content(monitor
       :content(PRICE ?x ?y))
```

```
(monitor
   :content (PRICE IBM ?price))
```

Figure 2 - KQML Examples

2.2 Agent Architectures

Many agent architectures have been developed to support multi-agent systems. Here we highlight a few to indicate the wide range of approaches that have been taken in developing such systems. The DECAF (Distributed, Environment-Centered Agent Framework) Agent Framework [4], developed at the University of Delaware, is a Java-based multi-agent system. It provides a matchmaker agent that accepts KQML performatives to allow for agent communcation. DECAF also provides a toolkit for building agents to remove the low-level details from the development process.

FIPA (Foundation for Intelligent Physical Agents) is developing a standard for multi-agent systems [5]. The standard includes specification for an agent architecture, agent management, agent communication, and interoperability. SRI International has developed an agent architecture called Open Agent Architecture (OAA) [6]. Agent interaction in OAA is done through an agent facilitator using a language called Interagent Communication Language (ICL).

Several agent architectures have been developed using CORBA (Common Object Request Broker Architecture) [7] as the underlying communication framework. CORBA provides seamless interaction among clients and servers in a distributed system. Using a CORBA framework to provide this functionality in a multi-agent system relieves the agent developer of providing low-level interagent communication. COBALT [8] is an agent framework based on KQML and CORBA. The implementation of KQML on CORBA provides a complete communication layer that can support cooperation in multi-agent systems. The system provides a mapping from the agent language KQML to the CORBA interface definition language (IDL).

Broadcom has developed the Agent Services Layer (ASL) as a layer on top of a CORBA framework [9]. The ASL provides services specific to agent interaction and coordination, and uses CORBA as the underlying distributed communication middleware. ASL allows agents developed in any language or platform to find and communicate with each other.

2.3 Related Real-Time Agent Work

There has been some recent work that considers how to handle real-time constraints in multi-agent systems. Early agent work at Stanford University [10] developed an architecture for real-time performance in intelligent agents. The work provides a framework that allows agents to trade quality for speed of response under dynamic goals, resource limitations, and performance constraints.

Work at the University of Massachusetts, Amherst [11] has developed a “design-to-time” scheduling algorithm for incremental decision-making. This algorithm provides a mechanism for making decisions

within specified timing constraints. For instance, when a specified deadline is near, the algorithm chooses a quicker decision-making method than if the deadline was longer. The DECAF architecture described above [4] has incorporated similar algorithms for agent scheduling.

At the University of Washington in St. Louis, the DOVE [12] project (Distributed Object Visualization Environment) uses a real-time agent framework based on TAO (The ACE ORB) [13] to provide a system for network management and performance visualization. This work is relatively new and the real-time agent framework is still in development.

The most recent draft of the FIPA standard specification [14] addresses real-time multiagent systems. The main issue that is considered is efficient coding of agent communication. Most agent communication is text-based, so that language commands have to be parsed on the receiving end. This is not appropriate in a real-time environment. The new FIPA draft specifies requirements for mechanisms for encoding communication among agents, efficient transport protocols, and time awareness within an agent framework. The Multi-Agent Coordination and Control special interest group of AgentLink [15] has called for the development of mechanisms for delivery of adequate responses to time critical methods and for efficient coordination methods.

3 A Real-Time Agent Architecture

In this section we define our model of real-time agents and of a real-time agent architecture. The agents in the system communicate through a real-time agent facilitator using an extension of KQML that allows for the expression of timing constraints. The architecture is designed with a real-time CORBA middleware layer intended for dynamic systems.

3.1 Model

A real-time agent is an agent whose actions may be time-constrained. That is, it must meet its stated goals under specified time constraints. As an example, consider the same data retrieval agent described in Section 2. However, now imagine that the agent exists in a system in which joint forces are collaborating in a battle management environment. This data retrieval agent may be tasked with retrieving satellite data from a ship on behalf of a nearby collaborating submarine. This application is time-critical in that the data should be retrieved quickly enough for critical battle decisions to be made. The real-time agent must coordinate with other agents in order to determine which data should be retrieved for the purposes of the pending decision.

A real-time agent architecture is a framework that provides a standard facility for expressing timing constraints on requests of real-time agents, and on communication among agents. The framework must also provide mechanisms for enforcement of these timing constraints. In our model, timing constraints are expressed through a real-time extension of the KQML agent communication language. Enforcement of timing constraints is handled by a real-time agent facilitator and an underlying real-time CORBA layer

3.2 Real-Time Agent Communication

Real-time agent communication involves specification of service capabilities and/or requirements through a real-time agent language, and functionality that allows agents to find and interact with each other.

3.2.1 Real-Time Agent Communication Language

Expression of timing constraints in the real-time agent architecture is handled through an extension to the KQML agent communication language. KQML performatives are extended with expressions of time for both specification of timing capabilities and for specification of timing constraints. Figure 3 extends the examples from Figure 2 with timing expressions on the KQML performatives. In the first example in Figure 3, one agent advertises that it can quote a stock price with an execution time of two seconds. Another agent asks the facilitator for service from an agent that can provide the price of IBM within three seconds. The third example in Figure 3 specifies that a requesting agent wants to monitor another agent that can quote the stock price of IBM every 3 seconds.

```
(advertise
   :language Prolog
   :content(monitor
       :content(PRICE ?x ?y)
   :exectime 2 seconds)
```

```
(ask-one
   :content (PRICE_IBM ?price)
   :receiver stock-server
   :language Prolog
   :deadline 3 seconds)
```

```
(monitor
   :content (PRICE IBM ?price)
   :period 3 seconds)
```

Figure 3 - Real-Time KQML Examples

3.2.2 Real-Time Agent Facilitator

The coordination of agents in the real-time agent architecture is managed by a real-time agent facilitator. The job of the real-time facilitator is similar to that of a typical agent facilitator. It finds an agent that best meets the specified requirements of the service requested by another agent. In the case of real-time agents, these specified requirements can include timing constraints, as illustrated in Figure 3 above. Further, the

real-time agent facilitator must also perform this task under its own timing constraints. That is, the search for the "best" agent to meet specified requirements must be limited in order to meet the deadlines, or other constraints of the agents involved.

For example, if an agent in the stock price application advertises with a KQML statement like the first example in Figure 3, the facilitator stores the information about this particular service. When an agent requests the service for getting the price of IBM, in the second example of Figure 3, the facilitator searches its database of registered services and finds the agent that can perform the service to meet the requirements of the requestor. This may require a schedulability analysis to determine if the service advertised by a particular agent can complete on its node in time to meet the timing constraints of the requesting agent.

3.3 Real-Time CORBA Agent Architecture

The real-time agent architecture is built upon a real-time CORBA layer that provides dynamic scheduling. This layer includes a scheduling service that dynamically schedules tasks based on earliest-deadline-first priority assignment. Whenever a new task enters the system, the scheduling service performs a schedulability analysis to determine if the new task can execute. By building the real-time agent architecture on top of this real-time CORBA framework, enforcement of timing constraints is handled by the underlying real-time CORBA services and the real-time ORB. Figure 4 displays the real-time agent architecture with the real-time CORBA layer between the agent communication layer and the real-time operating system.

The CORBA Trader Service [16] is a special CORBA object with a well-known name in the system that matches service requests with servers. A server registers with the Trader Service, indicating the services that it can provide. When a client requires a particular service, it can query the Trader Service with the requirements that it has for service, and the Trader finds the server that best meets its requirements. The University of Rhode Island and Mitre have developed a Real-Time Trader Service [17] that can find a server for a requesting client that can best meet all of its requirements, including its timing constraints. That is, the client specifies a deadline on the requested service, and the Real-Time Trader Service performs schedulability analysis to determine which server will be most likely to meet the timing constraints.

The real-time agent architecture realizes its real-time agent facilitator as an extension of this Real-Time Trader Service. The real-time agent communication layer of the architecture allows agents to communicate with the facilitator and with other agents using RT KQML. The real-time facilitator accepts service specifications and service requests in the form of RT KQML performatives. Between the real-time agent communication layer and the real-time CORBA services layer, the real-time KQML service request is translated to CORBA IDL for the real-time Trader. The real-time Trader finds the best agent to fulfill the

request. It must do this quickly enough to allow the serving agent to meet its timing constraints. This constraint is taken into account in the schedulability analysis of the request.

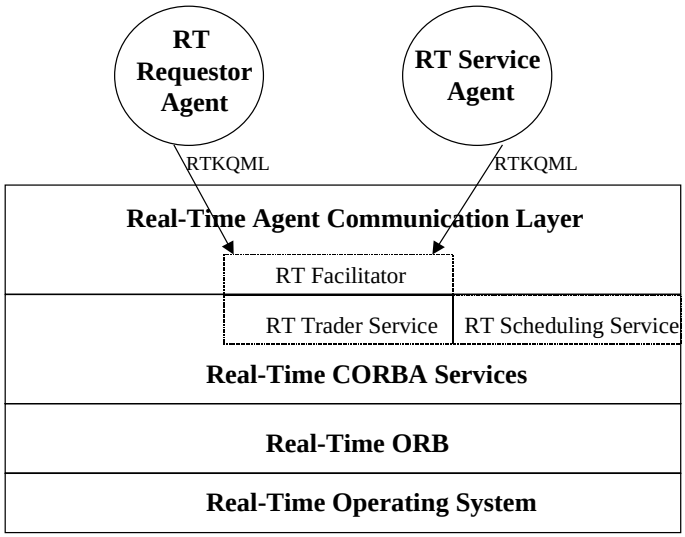

Figure 4 - Real-Time Agent Architecture

4 Conclusions and Future Work

This paper has presented a model for a real-time agent architecture that allows agents to work together to meet their specified design objectives in time-critical applications, such as electronic commerce, military training scenarios, and critical collaborative planning. The architecture provides a real-time agent communication language for expression of timing constraints and an underlying real-time CORBA layer that provides enforcement of timing constraints.

There are two levels Quality of Service (QoS) that can be explored in this model of a real-time agent architecture. First, as we have described in Section 2, there has been recent research towards providing differing levels of quality of response depending on the amount of time available [11]. The model that we have described in this paper could incorporate some of these algorithms and concepts in the execution of agent services. Individual agents could provide varying levels of accuracy, security, etc. depending on the timing constraints on the service it is performing. The real-time facilitator could use this type of algorithm to determine what techniques to use to search for the agent that best meets requestor's requirements.

A second level of QoS that could be explored in conjunction with the model described in this paper involves generalizing the model itself to express and enforce general QoS requirements. Timing constraints make up a subset of the kinds of requirements that can be specified in a general QoS model. We could extend the model described here to allow for the expression and enforcement of general QoS

requirements. This would require generalizing the kinds of constraints that could be expressed in the KQML language extension. For example, an agent could advertise that it can provide a certain level of security or accuracy, as well as timing capabilities. Requesting agents could specify requirements for accuracy, security and time. This generalization would also require extending the real-time CORBA model to allow for the enforcement of more general QoS requirements.

References

[1] Nicholas R. Jennings, Katia Sycara, Michael Wooldbridge. A Roadmap of Agent Research and Development. In *Autonomous Agents and Multi-Agent Systems*, 1, 275-306 (1998), Kluwer Academic Publishers, Boston.

[2] Keith Decker, Mike Williamson, Katia Sycara. Matchmaking and Brokering. In *Proceedings of the Second International Conference on Multi-Agent Systems* (ICMAS-96). Dec. 1996.

[3] Tim Finin, Richard Fritzson, Don McKay, Robin McEntire. KQML as an Agent Communication Language. In *The Proceedings of the Third International Conference on Information and Knowledge Management* (CIKM), ACM Press, November 1994.

[4] John Graham and Keith Decker. Towards a Distributed, Environment-Centered Agent Framework. In *Proceedings of the 1999 Intl. Workshop on Agent Theories, Architectures, and Languages [ATAL-99]*, Orlando, July 1999.

[5] FIPA. FIPA 98 Specification. 1998. http://www.fipa.org/spec/fipa98.html.

[6] David L. Martin, Adam J. Cheyer, Douglas B. Moran. The Open Agent Architecture: A Framework for Building Distributed Software System. *Applied Artificial Intelligence*. vol. 13, pp. 91-128. Jan-Mar 1999.

[7] OMG. Common Object Request Broker: Architecture and Specification. OMG Specification Revision 2.0, July 1995.

[8] D. Benech, T. Desprats. A KQML-CORBA based Architecture for Intelligent Agents Communication in Cooperative Service and Network Management. In *Proceedings of IFIP/IEEE International Conference on Management of Multimedia Networks and Services '97* July 8-10, 1997.

[9] Fergal Somers, Richard Evans, David Kerr. Scalable Low-Latency Network Management Using Intelligent Agents. *Communicate: Broadcom's Technical Journal*. vol. 3, issue 2. (http://www.broadcom.ie/communicate/vol_3_iss_2/papers2/FS.HTM).

[10] Barbara Hayes-Roth. Architectural Foundations for Real-Time Performance in Intelligent Agents. *Real-Time Systems*. May 1990.

[11] Alan Garvey, Victor Lesser. Design-to-time Real-Time Scheduling. *IEEE Transactions on Systems, Man and Cybernetics – Special Issue on Planning, Scheduling and Control*. vol. 23, no. 6, 1993.

[12] Douglas C. Schmidt. *The Distributed Object Visualization Environment*. http://www.cs.wustl.edu/~schmidt/dove.html.

[13] D. Schmidt, R. Bector, D. Levine, S. Mungee, G. Parulkar. TAO: A Middleware Framework for Real-Time ORB Endsystems. In *Proceedings of the 1997 IEEE Workshop on Middleware for Distributed Real-Time Systems and Services*, San Francisco, CA, Dec. 1997.

[14] FIPA. Real-time Multi-Agent Systems. FIPA99 Proposed Requirements Draft – Real-time ad hoc group. Draft Version 0.0, Jan. 1999.

[15] Agentlink. Special Interest Group in Multi-Agent Coordination and Control. Aug. 1999. http://www.agentlink.org/activities/sigs/sig5.html.

[16] OMG. *CORBAServices: Common Object Services Specification.* vol. 1. OMG Specification, March 1996.

[17] Steven Wohlever, Victor Fay-Wolfe, Bhavani Thuraisingham, R. Freedman, John Maurer. CORBA-based Real-time Trader Service for Adaptable Command and Control Systems. In *Proceedings of the Second IEEE International Symposium on Object-oriented Real-time Distributed Computing* (ISORC 99). May 1999.