Round-Table Architecture for Communication in Multi-Agent Softbot Systems

Pham Hong Hanh

Department of Computer Science
State University of New York at New Paltz
75 S. Manheim Blvd. Suite 6, NY 12561-2440
Fax: 1-914-157-3571
pham@mcs.newpaltz.edu

Tran Cao Son

Knowledge Systems Laboratory
Department of Computer Science
Stanford University
Gates CS Building, 2A, CA 94305
tson@KSL.Stanford.EDU

Abstract

In multi-agent systems based on softbots, communication architectures have significant influences on system performance as interaction and cooperation of softbots are carried out via agent communication. In this paper we propose a new communication architecture which is based on a round-table mechanism. Communication channels are preliminarily defined based on the matching of agent requests. A channel connects an agent to a queue of matched agents with the same interests and is scheduled to become periodically active based on their proportions in total demand and the amount of available resources. The order of activating channels and the sequence of agents in matched queues are defined based on agent time constraints. Our evaluation shows that the proposed model achieves a good balance of performance and quality of service compared with the other methods and is especially useful when the number of agents are very large and the capacity of systems is limited.

Keywords: multi-gent systems, architecture, communication, softbots, E-business.

1. Introduction

Agent technology is predicted as one of the most efficient tools to conduct business via Internet in an interactive, automatic, fast, and low-cost way [5,13,7]. Softbots are programs which can act autonomously to fulfill user tasks. In multi-agent systems, which are based on softbots, agents can be distributed on different hosts and interact and cooperate each other through communication. Thus, agent communication architectures have significant influences on system performance and quality of service.

The development of agent communication systems for agent-based software involves: (i) define formal languages for representing commands and the transferred information [13,19]; (ii) design communication architectures which include interaction mechanisms and communication models; (iii) develop a planning system for each agent [10,15,20] which defines when and what commands or information should be exchanged with other agents to achieve given goals. While there are many systems developed for (i) and (iii), (ii) receives less attention and yet to be developed.

In this paper we concentrate on communication architectures, i.e. (ii). In the next section, we discuss issues of communication in multi-agent softbot-based systems. Then, a new architecture for agent communication is described in section 3. To compare the proposed architecture with the others, an estimation is carried out in section 4. Finally, conclusion is given in section 5.

2. Communication in Agent-based Systems

General layout of a multi-agent softbot system and its communication management can be illustrated as in Fig. 1. Agent mobility, which is the ability of agents to move between hosts by themselves, has significant influences on the design of agent communication. There are three types in terms of agent mobility: Completely-Mobile, Semi-Mobile, and Non-Mobile, those review is provided in [16].

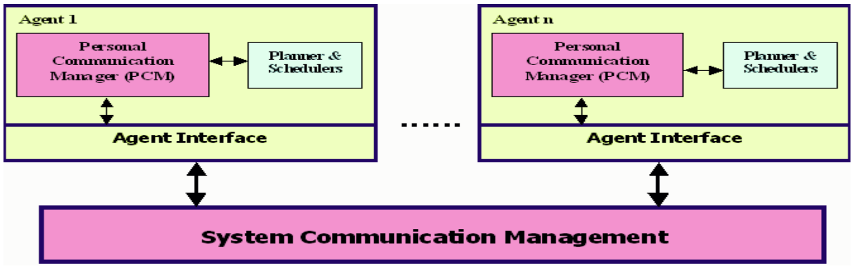


Figure 1

Requirements and goals in developing agent communication architectures can be stated as follows:

Given: n agents A1, A2, …, An with Q service-request categories R={R1, R2, …, R_Q}. Each agent Ai , i=1..n, characterizes by a set of data about the given agent Ai : (Ti, Si, Di) where,

- Ti is the period of time for which the agent is scheduled to live in the given system.
- Si shows about which services the agent would like to communicate with other agents. The deadlines for each service are also given in Si.
- Di is other data such as sizes of messages, message box's address etc.

Requirements: Design a model and mechanisms that define ways and orders by which agents {A1, A2, …, An} exchange information with the others based on their interests and needs given in Si, i=1..n. The goals are to reduce the message traffics in the system, especially to avoid system crush and agent starvation, while maintaining good quality of service such as response time, privacy, and customization.

Existing architectures for agent communication can be grouped into the following categories: *Yellow-pages* (**YP**)[3], *Contract-Net* (**CN**)[17], *Pattern-based* (**PB**)[15], and *Point-to-point* (**PP**) [11]. A study in [16] shows that most of them create communication channels using standard interprocess communication mechanisms and schedulers of low-level middlewares or operating systems, which are not designed to use other information about agents senders/receivers, such as their interests, deadlines of requests, agent life time. Therefore, these systems can support only very limited number of agents. When the number of agents increases, agent messages in these systems unfairly suffer from starvation [21]. Another critical issue in designing communication architecture is reliability. Most of these systems do not consider the limited capacity of the host systems and therefore would make systems crashed when the number of interactions is very high and overloads the capacity of the system, as reported in [1,11].

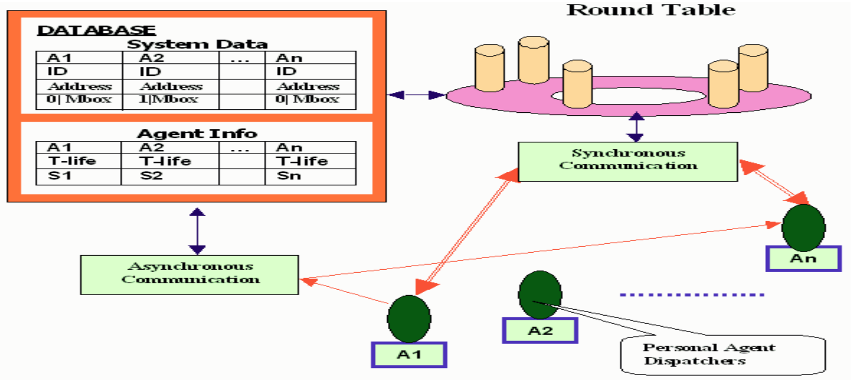
3. Round-Table Architecture

To overcome the mentioned shortcomings, we propose a new communication architecture for agent-based systems. This architecture can be used to manage agent communication in complete-mobile or none-mobile softbot systems. Our goals are (i) to take it into account the limited capacity of the host system and agent deadlines; and (ii) to achieve a good balance between the workload of agents and the workload of communication manager. We propose to have a combination of a centralized management and an autonomous management by each agent. Besides, the system resources such as memory and CPU's time will be divided fairly between agents and in an order according to the deadlines of requests and to agent live times.

3.1 Communication Model

Our model can be described as in Fig 2. The communication management involves (i) Database; (ii) Round Table; (iii) Agent Personal Dispatchers (built in each agent). System Data stores identification information, a pointer to a message box, and a flag of its status for each agent. For security, System Data can be accessed only by CCU and is not available for agents. Agent Data is formed at the registration when the agent enters the system. This data is accumulated based on the information submitted to the CCU during agent life. It has the following form:

- A_1: Life time T_1, service interest $S_1=\{(R_1^{1}, t_1^{1}), \ldots,(R_1^{D1}, t_1^{D1})\}$; …
- A_n: Life time T_n, service interest $S_n=\{(R_n^{1}, t_n^{1}), \ldots,(R_n^{Dn}, t_n^{Dn})\}$.



A1	A2	…	An
ID	ID		ID
Address	Address		Address
0| Mbox	1|Mbox		0| Mbox

A1	A2	…	An
T-life	T-life		T-life
S1	S2		Sn

Figure 2 Communication Management Components

An agent communicates with the others by sending messages. The contents of messages are defined by agent planning systems whose purposes could be: *searching*; *cooperation*; *trading* and *negotiation*. Here we focus on how to manage messages, which is how to build environment and means for communication rather than the context of communication, which is studied in agent planning. The communication management is carried out by agent personal Dispatchers (APD) and CCU which provide agents two alternatives of communication (Fig.3): (i) Synchronous; (ii) Asynchronous. In asynchronous mode, an agent X sends a message directly to a target agent at any time when he needs. This message is stored in the receiver Y's message box. In synchronous mode, an agent can use services of the Round Table to create a communication channel to a queue of agents who would meet his interests. The protocols for synchronous communication are described in details in [18]. First, the agent sends a request, which contains data about his interests in the message body. The CCU processes the agent requests to define the matched queue. Then, the seat for an agent in the Round Table is defined by his own Dispatcher. Next, a permanent communication channel is automatically established between the agent and a queue based on agent seat and the rules of the Round Table, which are described in the next section. Since then, this agent will send/receive messages synchronously within a given period of time Dts which is defined by the Round Table. Algorithms for APD and CCU in synchronous communication are described in [18].

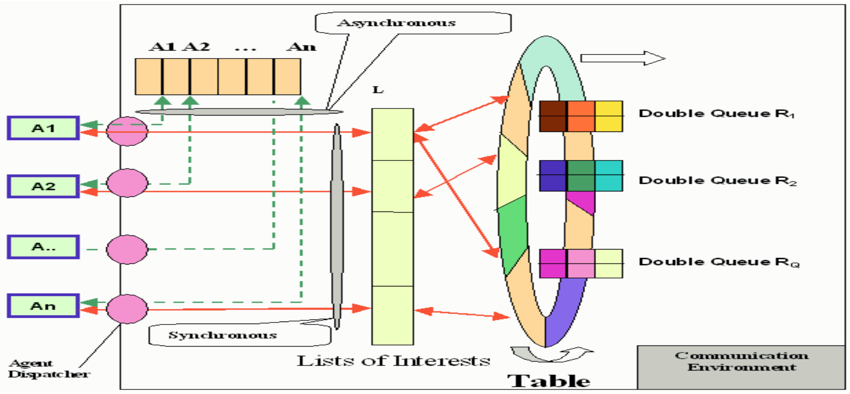


Figure 3. Round-Table for Agent Communication

3.2 Structure of Round Table

Round Table is a mechanism which maps agents according to their interests and then creates communication channels between the matched ones. Unlike other communication models, communication channels in this model are established with consideration of agent time constraints. Round Table also controls the number of channels based on the available resources (threads and memory). Round Table has Q double queues and virtually a chain of seats. The Q queues $\{R_1, R_2, \ldots, R_Q\}$ are formulated based on the agent interests in Si, i=1..n. In each double queue Rj, j=1..Q, we have two subqueues: $R^+(j)$ and $R^-(j)$ which are a list of agents who are interested in providing service $R^+(j)$, and a list of agents who are interested in demanding service $R^-(j)$ respectively.

- $R^+(j)=\{\{A^1, t_j^1\}, \{A^2, t_j^2\}, \ldots \{A^{u(j)}, t_j^{u(j)}\}\}$
- $R^-(j)=\{\{A^{1*}, t_j^{1*}\}, \{A^{2*}, t_j^{2*}\}, \ldots \{A^{u(j)*}, t_j^{u(j)*}\}\}$

Where, $A^k \in AS=\{A_1, A_2, \ldots, A_M\}$, k=1…a(j) or k=1…a(j)*, is a set of agents who use Round Table services; t_j^k is the time constraint for the given agent request concerning Rj service, either in providing or demanding.

If each agent, who wants to use the Round Table for synchronous communication, has an entry to the Round Table then we would need M entries to the

Round Table at a time. Each entry has its queue of requests which is a list of interests of the given agent, for instance for agent A1 it could be L(1)={R1+,R2-, R3+,…} . This list is maintained by the APD and is sorted based on the time constraints of the interests given in S_1. Assume that Mc is the maximal number of channels, which can be created using the available resources. Then, the total number of seats in the Round Table is Mc. These seats are distributed to the agents by the following law: for each category Rj of services, i.e. for each double queue, the number of seats for agents who are interested in Rj is defined as follows:

$$\Delta j = \frac{Mc \times Kj}{\sum_{i=1}^{Q} Ki}$$, where Kj, j=1..Q, is the number of agents interested in Rj.

Thus, Nc channels would be given to Q services by the following rule:

$$Mc = \sum_{j=1}^{Q} \Delta j = \sum_{j=1}^{Q} \frac{Mc \times Kj}{\sum_{i=1}^{Q} Ki}$$

If a service is represented by a sector in the Round Table then we have Q sectors and each sector SR[j], j=1..Q has Δj seats and thus can provide channels Δj for agents who are interested in service Rj at a time. According to the protocol described in Fig.11, each agent receives the list of sectors with their size. This list shows the agent which queues would probably meet his needs. Given that the agent Ai has a list of request $L^{i}_{[h]}$, h=1..Hi, i=1..M. The agent Dispatcher would define by himself the sectors which is best fit his interests and time constraints. Then, agent sends a request to CCU for a seat in the given sector for each $L^{i}_{[h]}$. Receiving this request, CCU checks if there is a free seat in the given sector. If so, CCU proves agent request. If no more seat is available CCU creates a waiting queue for the given sector and assign requests from the waiting queue to the available seats according to the Priority of the requests. The priority of a request $L^{i}[h]$ can be defined as: $PL^{i}[h] = F(Ti, t^{i}_{h}) - Age(Ti)$ where, *F* is some function defined by CCU ; *Ti* is life time of agent Ai; *Age(Ti)* is an aging function which increases priority of an agent by the time the agent name is in the system. We use this technique to avoid agent starvation; t^{i}_{h} is the deadline of request $L^{i}[h]$ of agent Ai.

Thus, for each Rj service category, Δj channels are given to agents who have shortest life times. After defining seat for an agent request $L^{i}_{[h]}$, for instance an offer of service Rj, the Round Table creates a synchronous communication channel between agents having seats and the matched subqueue $R^{+}(j)$. After that, the given request which is currently ranked highest L(i), with Hi position, is removed from the subqueue top. Synchronous communication channel starts with suggesting Ai to the agent listed in the head of the subqueue $R^{-}(j)=\{\{A^{1*}, t_j^{1*}\}, \{A^{2*}, t_j^{2*}\}, \ldots \{A^{u(j)*}, t_j^{u(j)*}\}\}$, i.e. A^{1*}, and then the next A^{2*}, etc. The queues are rounded backward, after $A^{u(j)*}$ the next one will be A^{1*} again. For each suggestion A^{k*}, agent Ai can choose to: exchange massages; skip and go to the next; go back to the head of the queue. The order of potential target agents in a subqueue is defined by their priorities as the following: $PTA^{j-}[k*] = G(Ta^{k*}, t_j^{k*})$ where, *G* is a Round-Table function; Ta^{k*} is life time of agent A^{k*}; t^{k*}_{j} is the deadline of agent A^{k*} interest in Rj ; Algorithm of Round Table mechanism in synchronous communication is described in [18].

4. Comparative Evaluation

In order to estimate the performance of the proposed model, compared with the other existing ones we use the following criteria: (i) *Cost of EC: Cn* – the time complexity spent for establishing communication network, usually for matching agents and filtering messages; (ii) *Maximal Number of Channels: Mc* - the possible highest number of channels in the agent communication system at a time; (iii) *Density: Dn* - the maximal average number of channels to/from an agent.Performance characteristics of PP, PB, CN, and YP methods, described in [16], and of the new architecture are shown in Table 1.

Table 1. Performance Characteristics

Methods	***Cost of EC***	***Maximal Number of Channels***	***Density***
Point-to-point	0	$(n-1)n/2$	$(n-1) \rightarrow m^*$
Pattern-based	$Q \times n$	$Q \times n$	$Q \rightarrow m^*$
Contact Net	0	$n.$	1
Yellow pages	$Q \times n$	$n.$	1
Round-table	$Q \times n$	Mc	$(Q/2+1) \rightarrow m^*$

(m* is the number of agents matched the requests of the given agents)

We use a set fuzzy values {VL,L,M,H,VH} stand for {Very Low, Low, Medium, High, Very High} and the following criteria for measuring quality of service for the given communication methods: Agent workload; Agent response time; Privacy; Customization (Flexibility). A comparison of quality-of-service of PP, PB, CN, YP, and RT architectures, which have been analyzed in [16], is shown in Fig 4.

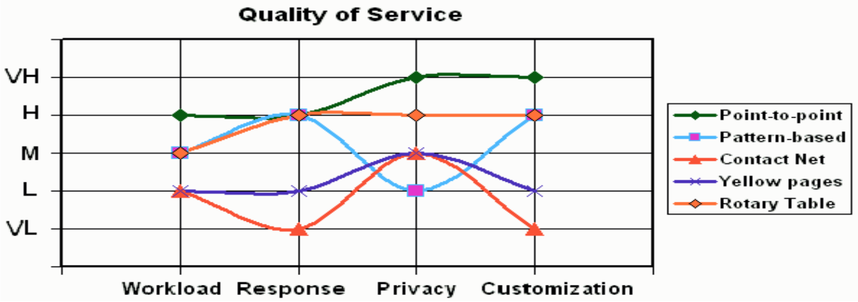


Figure 4. Fuzzy Comparison of PP, PB, CN, YP in term of Quality of Service

Notice that in our system each agent can have both synchronous and asynchronous communication at the same time. This feature gives agents more freedom and flexibility as they can first use synchronous communication with the Round Table mechanism for a brief look at the e-market, having short conversations with the potential agent-partners from the selected matched queues. Then, each agent can use his thinking engine to negotiate and define the "right" partners. Next, the agent can exploit asynchronous communication as they already know the target partners. The asynchronous communication is good for one-to-one negotiation, while synchronous communication with Round Table mechanism would be very suitable for surveys.

5. Conclusion

We have proposed a new architecture for agent communication which considers the system and time constraints and is able to scale itself to adapt to the limitation including the change of system capacity. Thus this, communication architecture would especially be useful in massive agent-based systems which have many agent-based applications running on hosts with limited resource. Our analysis and evaluation show that the proposed Round-Table communication architecture is also flexible and achieves a good balance of system performance and quality of service. In the future we intend to embed the given architecture into an e-business system for mobile services which is proposed in [23] by VTT Electronics of Finland.

Acknowledgment

This work is supported by the Research Creative Award N 00-01 from the State University of New York at New Paltz.

References

[1]. Arnal M. and Faltings B., Smart clients: Constraint satisfaction as a paradigm for scalable intelligent information systems, AAAI Workshop on AI for E-commerce, p10-15, 1999.

[2]. Beck M. et al, Active and Real-time Functionality for Electronic Brokerage Design, *1^{st} WECWIS*, 1999.

[3]. Bradshaw J. et al, Agents for the Masses, *IEEE Intelligent Systems*, p53-63, March-April 1999.

[4]. Buttazzo G. and Sensini F., Optimal Deadline Assignment for Soft Aperiodic Tasks in Hard Real-Time Environment, *IEEE Tran. On Computers*, V48-N10, p1035-1052, 1999.

[5]. Cynthia McFall, An object infrastructure for Internet Middleware IBM on Component broker, *IEEE Internet Computing*, March-April 1998, p46-p51.

[6]. Dabka Padmanabh, Enterprise Integration via COBRA-based Information Agents, *IEEE Internet Computing*, p49-57, Sep-Oct 1999.

[7]. Elofson G. and Robinson W., Creating a Custom Mass-Production Channel on the Internet, *ACM Communications*, p56-62, March 1998.

[8]. Henning M., Binding, Migration, and Scalability in CORBA, *ACM Communications*, p62-71, October 1998.

[9]. Hiltunen M. et al. Real-time Dependable Channels: Customizing QoS Attributes for Distributed Systems, *IEEE Tran. On Parallel and Distributed Systems*, Vol. 10, N6, p600-611, June 1999.

[10]. Jamali J., Thati P., and Agha G., An Actor-Based Architecture for Customizing and Controlling Agent Ensembles, *IEEE Intelligent Systems*, p38-44, March-April 1999.

[11]. Kiniry Joseph and Zimmerman Daniel, A Hands-on look at Java Mobile agents, *IEEE Internet Computing* July-August1997.

[12]. Knapik Michael and Johnson Jay, *Developing intelligent Agents for Distributed Systems*, McGraw-Hill, 1998

[13]. Labrou Yannis, Tim Finin, and Yun Peng, Agent Communication Languages: The Current Landscape, *IEEE Intelligent Systems*, p45-52, March-April 1999.

[14]. Lange D. and Oshima M., Seven good reasons for Mobile agents, *ACM Communications* March 1999, p88-89

[15]. Ma Moses. Agents in E-commerce, *ACM Communications*, p79-80, March 1999.

[16]. Pham H. Hanh, Nguyen Hien, Nguyen V.Hop, Environment and Means for Cooperation and Interaction in E-commerce Agent-based Systems, proceeding of the International Conference on Internet Computing (IC'2000), Las Vegas, June 26 - 29, 2000.

[17]. Pham H. Hanh, Worasing Rinsurongkawong, and Thanadee Ujjubandh, "Distributed Multi-level Adaptation for Dynamic Multi-Agent Systems", in the Proceeding of 22nd *IEEE Conference on Systems, Man, and Cybernetics*, SMC'99, Tokyo, Japan, October-1999.

[18]. Pham H. Hanh, Agent Communication, TR-2000-06-01Tech. Report, SUNY at New Paltz, 2000.

[19]. Reeves D. Grosof B., Wellman M., and Chan H., Toward a Declarative languages for Negotiating Executable Contracts, *AAAI Workshop on AI for E-commerce*, p39-45, 1999.

[20]. Smith R.G., The contract Net protocol: High-level Communication and Control in a Distributed problem Solver, *IEEE Tran. On Computers*, V29-N12, p 1104-1113, 1980.

[21]. Wong D., Paciorek N., Moore D., Java-based Mobile Agents, *ACM Communications*, p92-105, March 1999.

[22]. Yamamoto G. and Nakamura Y., Architecture and Performance Evaluation of a Massive Multi-Agent System, Autonomous Agents'99, Seattle USA, pp.319-325, 1999.

[23]. Palola M, Heikkinen, Constructing Mobile Web Services on a software agent platform, proceeding of the International Conference on Internet Computing (IC'2000), Las Vegas, June 26 - 29, 2000.