RAMONAITE A., HUTZLER G., "Multi-Agent System for Internet Acceleration", in *Proceedings of Starting Artificial Intelligence Researchers Symposium STAIRS 2002*, pp. 35-44, IOS Press, 2002.

Multi-agent System for Internet acceleration

Ausra RAMONAITE, Guillaume HUTZLER
LaMI – UMR 8042, CNRS / Université d'Evry Val d'Essonne
523, place des Terrasses, 91000 Evry, France

Abstract. In recent years the study of multi-agent systems has become an active research field in Artificial Intelligence. The exponential growth of World Wide Web has revealed new opportunities for the development of these systems in the electronic commerce, the information searching and recently the Internet acceleration. In this article we present a multi-agent system, called BoWeMA, for data flow acceleration through the Internet. Its distributed architecture, based on network acceleration concepts, includes data flow analysis, optimisation, compression and dynamic memorisation. The proposed communication model, complying with FIPA ACL standard, offers flexible and efficient messaging between the agents, which is very important in this Internet solution.

1. Introduction

Multi-agent system (MAS) [10] can be defined as a distributed system, in which a certain number of homogenous or heterogeneous entities, called agents, work according to the complex modes of interactions, in order to perform their individual tasks as well as to reach the desired global objectives. In recent years the study of MAS has become an active field of research in Artificial Intelligence. A great number of MAS applications include control systems, industrial, e-commerce and medical applications, computer simulation, telecommunication systems, etc. The exponential growth of the World Wide Web has revealed new opportunities for multi-agent systems development in the Internet acceleration. Due to rising data flow volumes, richer content types and increasing user expectations, the Internet has become one of the bottlenecks to network performance.

Our research aim is to design a complete solution for data flow acceleration through the Internet, including the following principles: (i) distribution, to avoid the bottlenecks of the system, (ii) cooperation between the different elements of the system, to avoid their overlapping and reworking, and (iii) communication between these elements, to ensure the coordination of their work. In this context, the multi-agent technology offers appropriate concepts to reach this objective. In this article we present a distributed multi-agent architecture, called BoostWeb Multi-Agent (BoWeMA), for data flow acceleration through the Internet. The proposed communication model, based on the specifications of FIPA ACL, offers efficient and flexible messaging between the agents.

The paper is structured as follows: in section 2 we briefly present related works. In section 3 we describe the main characteristics of the BoWeMA system, in particular the distributed multi-agent architecture and the communication model, which complies to the FIPA ACL specification. Finally, in section 4 we discuss some concluding remarks and future work.

2. Related Works

The continuous growth of the information available online and the world-wide expansion of the Internet have introduced new issues such as time-consuming access to the information of any nature and bandwidth insufficiency. The first attempt at addressing these problems

consists of infrastructure upgrades [3]: installing more powerful computer hardware or faster network equipment. Hardware upgrading provides some flexibility, enabling faster access to the HTTP servers, but it does not eliminate the network bottlenecks, and it is also costly and complex to apply to extended networks.

Another solution, addressing the problem of Internet performance, includes content replication and load balancing [9], by adding multiple local servers to a central site [7] or by distributing Web servers over several geographical locations [8]. These methods reduce the load on the central Web server, and thus the time it takes to respond to a navigator request. However they present some limitations: (i) the problem of the latency associated with other network equipment such as routers and transmission devices, is not solved; (ii) providing consistent content and managing multiple servers, especially when they geographically distributed, is complex and costly.

The caching technique [2, 4, 16] is one of the ways to reduce the overhead of generating the Web documents. The cache located near the requester, works to accelerate data delivery and to optimise bandwidth utilisation, when the requested data is stored in the cache. However this technique presents the following limitations: (i) voluminous files and a great number of multimedia objects cannot be cached; (ii) when memorising changing Web documents, the cache coherence becomes difficult to control.

Global network optimisation employing software tools can also enhance Internet performance, reducing the end-to-end HTTP traffic in the system. Such optimisation tools give immediate results by accelerating connection and response times and by decreasing the data traffic volume. BoostWorks [1] has developed a software product, called BoostWeb Optimizer [6] that accelerates Web page delivery and reduces the data traffic on the network. BoostWeb boosts network performances by taking advantage of the high volume of redundant data flow on the Internet. This technology is based on Intelligent Network Acceleration (INA) architecture. It analyses the data, passing through the network, optimises them and applies the most adapted compression, while taking into account the capabilities of the end-user. BoostWeb Optimizer is easy to deploy and requires no changes to Web servers or applications. However it presents some limitations: (i) the different functionalities of data processing (for example, the data compression and optimisation) are carried out in a centralised way which may cause bottlenecks when the long data processing is required; (ii) the lack of verification of the efficiency of data compression may result in a cost of compression for the user which is greater than the expected gain.

Taking into account BoostWorks' INA concepts, we introduce in this paper a distributed multi-agent architecture BoWeMA that allows the simultaneous processing of a great number of navigator requests and to obtain a better regulation of the data flow by avoiding the system bottlenecks during intensive data processing. To achieve this, we use a collection of agents having a certain number of specialised competencies required for data flow processing. These agents should behave collectively in order to equally share the work each time when a new navigator request is received.

3. BoWeMA: Distributed Multi-agent Architecture for Data Acceleration through the Internet

The BoWeMA (Boost-Web Multi-Agent) system is designed to provide the interface between a customer - the data user (for example, the Web navigator) and a server - the provider of these data (for example, the HTTP server). It analyses the data exchanges between the user and the server in order to optimise the redundant HTTP flow. With the BoWeMA system, we propose a distributed multi-agent architecture which avoids the data redundancies and should accelerate the information flow to and from the HTTP server by

applying an "intelligent" and distributed processing (analysis, compression and optimisation) to the delivered data.

3.1. General Architecture

The BoWeMA architecture is comprised of two layers (Figure 1): the cognitive layer, integrating Intelligent Agents (Specialised Agents and Passive Administrators) that analyse the received data flow and apply their distributed and "intelligent" processing; and the reactive layer, containing Code-Agents , whose main task consists in executing predefined actions such as data flow compression or optimisation.

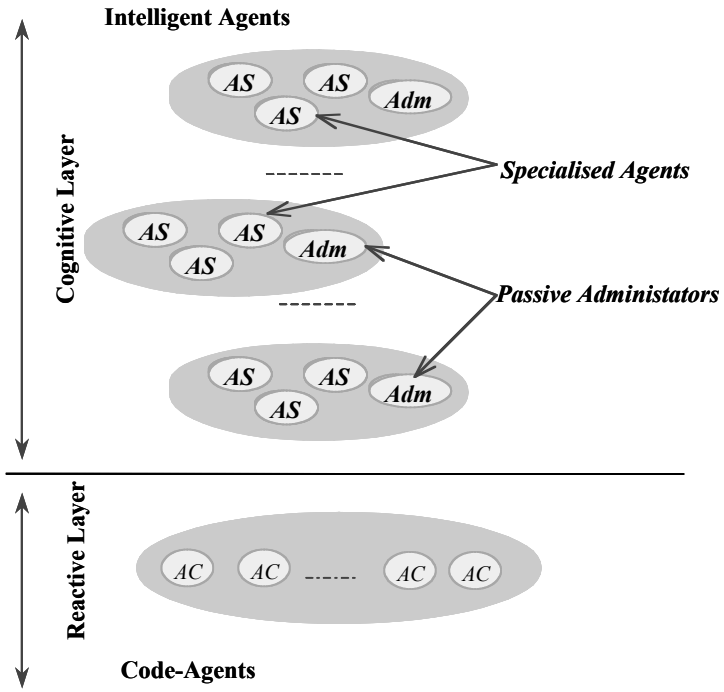

Figure 1. The general architecture of BoWeMA

The Specialised Agents are divided into several groups according to the specificity of the tasks they have to handle. Each group has its Passive Administrator. This administrator tracks information about changes in the agents' states in a group (the agents may be occupied or unoccupied). This information is transmitted by the agents themselves. With this available information, the Passive Administrator manages the occupation of a group, by creating new agents when all the agents of a group are occupied, or by suppressing some of them when the workload is low.

3.2. Organisational Structure

For each navigator request concerning some Web page, the corresponding processing requires the cooperation of the different agents having varied competences and transmitting mutually the requests or the data in order to satisfy the initial navigator request. We identified a succession of elementary processing actions, where the specialised agents analyse the data flows more and more precisely until the final determination of the type of compression or optimisation agent needed.

The organizational structure of BoWeMA corresponds to different roles, attributed to the intelligent agents according to the different tasks they have to handle. The following roles are defined (see Figure 2):

- *Data Flow.*
  The main task of the Data Flow Agents (referred to as *AF* in Figure 2) is to identify the nature of received data flow (the navigator request or the HTTP server response complying to HTTP/1.1 protocol [11]) and to reroute it to the corresponding agent or the system actor (the navigator or the HTTP server). The navigator requests include ordinary requests containing no data (for example, the request for loading a certain Web page) and requests containing data (for example, a request for updating a certain Web page). The HTTP server responses include ordinary responses containing data such as the Web pages requested by the navigator, and responses containing no data that may correspond to the negative HTTP server answers concerning some requested Web pages.
- *Analysis, including two sub-roles: Request Analysis and Request-Response Analysis.*
  The Request Analysis Agents (AA^{req}) analyse the ordinary requests, received from the navigator. Their main task is to verify if the information requested by the navigator is stored in the cache. The Request-Response Analysis Agents (AA^{rr}) analyse the navigator requests or the ordinary responses, received from the HTTP server. They identify the type of received data flow (HTML, JPEG, GIF, etc.) and determine the associated data processing.

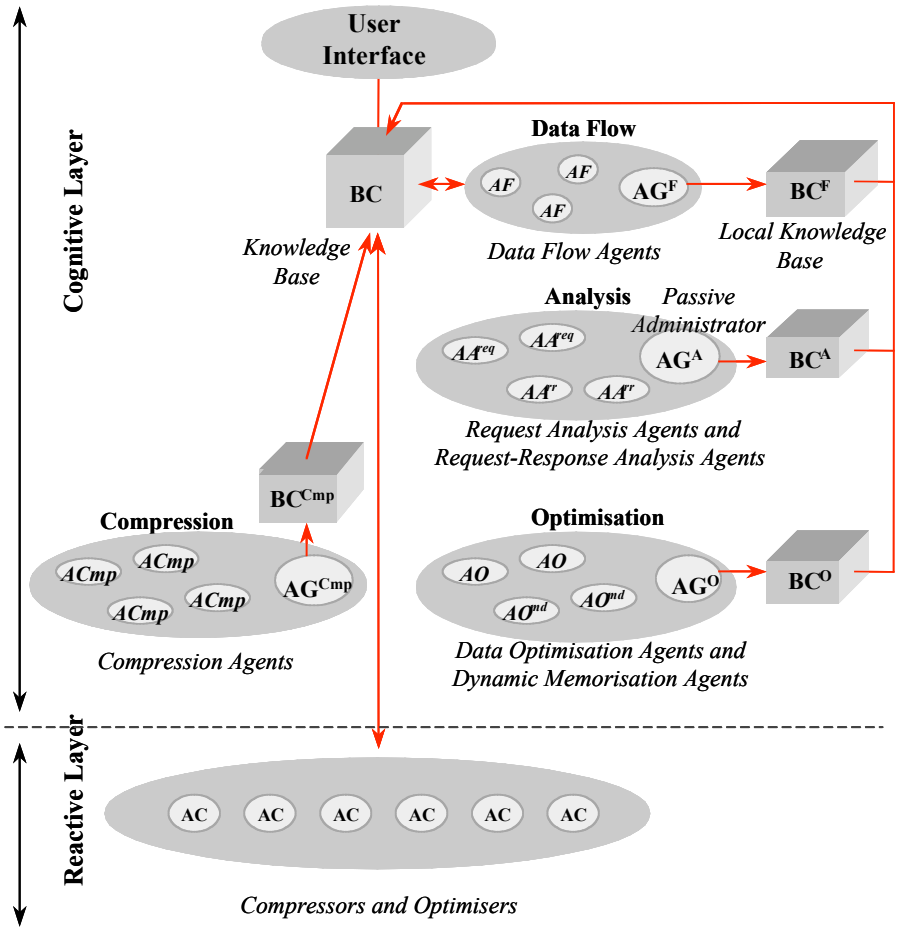

Figure 2. The organisational structure of BoWeMA

- *Optimisation, including two sub-roles: Data Optimisation and Dynamic Memorization.*
  The data optimisation aims to obtain more homogenous data flows, eliminating insignificant and redundant elements and preserving the original data format. The dynamic memorization stores the most frequently requested data (in its original and optimised versions) in the cache so that it can quickly deliver up-to-date information to the navigator. The Data Optimisation Agents (*AO*) identify the type of received information (text, image, etc.) and reroute it to the corresponding optimiser (the code-agent, able to optimise a certain type of data). The main task of Dynamic Memorization Agents (AO^{md}) is to place the data in its original and optimised versions in a database, which plays the role of cache.
- *Compression.*
  The data compression includes the compression of text or binary files as well as the intelligent reduction of image (GIF, JPEG, PNG) size with no visual loss of content. The Data Compression Agents (*ACmp*) identify the type of received information (the text file, the image, etc.) and reroute it to the corresponding compressor (the code-agent, which is able to compress that type of data).

3.3. Multi-agent communication model

In the BoWeMA architecture we consider the communication as an action of information exchange between the agents by sending the messages peer to peer. We identify two types of communication:

- "Vertical" communication between the specialised agents from the different groups, having the different levels of competence (for example, between the Data Flow Agent and the Request Analysis Agent) or between the specialised agents and the code-agents (for example, between the Data Compression Agent and the code-agent, called Compressor);
- "Horizontal" communication between the specialised agents and their passive administrators (for example, between the Data Flow Agent and its Passive Administrator).

For the agents to communicate effectively, a common communication language is required [15]. In the field of the Internet acceleration the heterogeneous agents should be able to exchange standardized messages in a flexible way. They should react quickly without wasting too much time on the communication actions themselves. In the BoWeMA system, all the agents communicate sending asynchronous messages that comply to the FIPA ACL specification [12, 13]. Having semantics that is clear and independent from the agents' structure, FIPA ACL allows setting up primitive and complex interactions, requiring the establishment of elaborate protocols.

3.4. Example of Data Transfer Procedure through the Internet using BoWeMA

To clarify the interactions between the different agents in the BoWeMA system, we present the following example: the navigator sends an ordinary request for loading a Web page. The BoWeMA system receives this request and transmits it to the HTTP server. After receiving, in return, the requested data (Web page), the BoWeMA provides the corresponding processing and transmits the result to the navigator.

The example described above, is composed of the following steps [17]:

- *Step 1: Receiving the navigator request.*
  The navigator opens the connection towards the BoWeMA system and sends the request back (see Figure 3 (1)). This request is received by one of the unoccupied Data Flow Agents (*AF*) that informs the Passive Administrator (AD^F) of the group about the changes in its state (Figure 3 (2)). Then the Data Flow Agent analyses the received information and determines the appropriate processing to apply to this request. In this case, the ordinary

request is supposed to be sent by the navigator. Therefore, the Data Flow Agent reroutes it to one of the unoccupied Request Analysis Agent (AA^{req}) (Figure 3 (3)) and once more informs the Passive Administrator about the changes in its state.

It should be noticed that all specialised agents inform their passive administrators about the changes in their states in the following cases: (i) after receiving the information sent by another agent or a system actor (Web navigator or HTTP server), and (ii) after sending the processed information to another agent or a system actor.

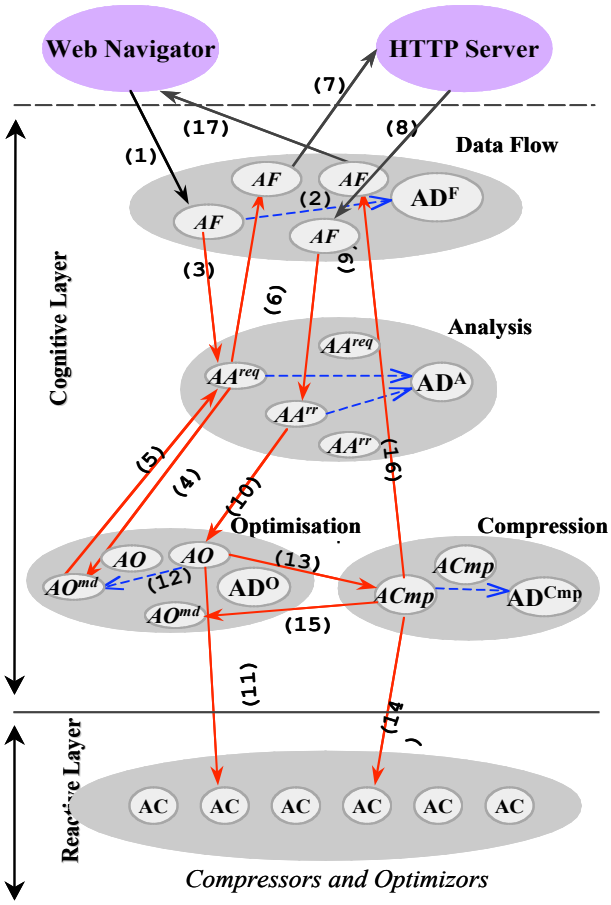

Figure 3. Interactions between the different agents in BoWeMA (solid-line arrows correspond to the interactions between the agent from the different groups; dashed arrows show the interactions between the agents from the same group)

- *Step 2: Request Analysis.*

After receiving the ordinary request rerouted by the Data Flow Agent, the Request Analysis Agent (AA^{req}) verifies if the requested data is already stored in the cache. To this end, it sends a request to one of the unoccupied Dynamic Memorisation Agents (AO^{md}) (Figure 3 (4)). The Dynamic Memorisation Agent replies by sending either the data in its optimised version corresponding to the navigator request or a negative answer if the requested format of data is not found in the cache. In this case, we will consider that the cache does not contain the requested information. Therefore the Request Analysis Agent receives from the Dynamic Memorisation Agent the negative answer (Figure 3 (5)) and reroutes the navigator request to one of the unoccupied Data Flow Agents (*AF*) (Figure 3 (6)).

- *Step 3: Sending the request to the HTTP server.*
  As the requested data is not stored in the cache, the Data Flow Agent (*AF*) opens the connection towards the HTTP server and sends the navigator request to it (Figure 3 (7)).
- *Step 4: Receiving the HTTP server response.*
  Answering to the navigator request, the HTTP server sends to one of the unoccupied Data Flow Agents either the requested data or the negative answer if this data is not available (Figure 3 (8)). In this case, we will consider that the requested data (Web page) is available, so the Data Flow Agent reroutes it to one of the unoccupied Request-Response Analysis Agents (AA^{rr}) (Figure 3 (9)).
- *Step 5: Analysing the response.*
  After receiving the response from the Data Flow Agent, the Request-Response Analysis Agent carries out several data analysis actions (for example, the verification of the type of received information, the verification of navigator capacities to decompress the corresponding type of data, etc.) in order to decide on the appropriate processing of the received information. In this case, the HTTP server response includes the HTML page, and the navigator is able to decompress such type of data. Therefore, the Request-Response Analysis Agent transmits the response to one of the unoccupied Data Optimisation Agents (*AO*) (Figure 3 (10)).
- *Step 6: Processing the response.*
  When the Data Optimisation Agent receives the data, it analyses this information selecting the best optimisation and/or compression agents. Then it contacts the corresponding optimiser (code-agent) asking to optimise the corresponding data (Figure 3 (11)). If the data can not be compressed, the Data Optimisation Agent sends a copy of the optimised data version to one of the unoccupied Dynamic Memorisation Agents (AO^{md}) asking to store it in the cache (Figure 3 (12)) and reroutes the optimised data directly to the Data Flow Agent. Otherwise, it sends the optimised information to one of the unoccupied Data Compression Agents (*ACmp*) (Figure 3 (13)). This agent contacts the corresponding compressor (code-agent) asking to compress the data (Figure 3 (14)). Then, the Data Compression Agent sends a copy of the compressed data to one of the unoccupied Dynamic Memorisation Agents asking to store it in the cache (Figure 3 (15)) and then reroutes the compressed information to one of the unoccupied Data Flow Agents (Figure 3 (16)).
- *Stage 7: Sending the response to the navigator.*
  After receiving the response, the Data Flow Agent (AF) looks for the navigator that has requested this information, and sends the response to it (Figure 3 (17)).

3.5. Discussion

We explained the different interactions between the agents in the BoWeMA, by analysing the example of a Web page loading. In this section we summarize several observations.

It can be seen that the distribution of data flow between the specialised agents of the same group as well as the distributed data treatment (data analysis, compression and optimisation), carried out by the specialised agents from the different groups, help to avoid the bottlenecks of the system, especially in the cases of intensive data processing. For example, the navigator sends several requests for loading various Web pages. These requests are received by several Data Flow Agents. After rerouting the received data flow to the corresponding agents (Request Analysis Agents) and informing their passive administrator about the changes of their states, the Data Flow Agents, mentioned above, become unoccupied. They can receive new data flows from the navigator while the Request Analysis Agents continue the processing of the previous requests. By the way, the complex data flow (for example, a Web page including the text and several images) can be distributed between the agents of the same group (for example, the Data Flow Agents). Such distribution accelerates the data treatment at

the corresponding level of BoWeMA (Data Flow, Analysis, Optimisation or Compression) as well as the release of the agents of this level for the processing of new data flows.

The second observation concerns the evolution of the number of agents in the BoWeMA, which helps to prevent the bottleneck problems which occur when all the agents of the same group are occupied. Upon initialising the BoWeMA application, each group of Specialised Agents has a certain number of agents. Each agent informs the passive administrator of a group about the changes in its state (occupied or unoccupied). Taking into account the occupation of a group, the passive administrator may create some agents in order to avoid the "overload" of a group. It should however be noticed that the total number of agents in a group is limited, as too many agents may injure the system performance.

We developed a prototype of the BoWeMA system, using the FIPA-compliant agent-based platform JADE [5, 14] (Java Agent Development framework), which is an agent framework that supports a rich set of agent platform services and provides the middle-ware necessary to manage the agents as well as some tools that simplify agent development and system debugging. This implementation has enabled to confirm the basic assumptions that we made at the beginning of our work.

4. Concluding Remarks

In the context of an exponential growth of exchanged data over the Internet, it is of fundamental importance to reduce:

- the number of useless transfers between HTTP servers and end-users (using cache techniques);
- the volume of the exchanged data (using compression techniques);
- the latency (using techniques that accelerate the processing of requested data).

The BoostWeb Multi-Agent architecture presented in this article address more specifically this last point. By distributing the treatment over a variable number of agents, several goals can be reached. First of all, several requests may be handled simultaneously, and single requests may themselves be decomposed into sets of elementary treatments, handled by separate agents. This way, the different elements of Web pages can be delivered asynchronously as agents finish their jobs. Secondly, the dynamic management of agent groups (i.e. dynamic creation/deletion of agents) allows the system to be very robust and adaptive in a context in which it is difficult to make any prediction about the future load of the system.

These characteristics open the way for a great number of applications concerning Internet development, and more generally, data exchanges in the telecommunication networks.

References

[1] http://www.boostworks.com.

[2] Ch. Aggarwal, J.L. Wolf and Ph.S. Yu, Caching on the World Wide Web, *IEEE Transactions on Knowledge and Data Engineering* 11/1 (1999) 94-106.

[3] R.Alliane, Des composants intelligents pour l'accélération des réseaux: modélisation et spécification. Master's thesis. Univ. of Evry Val d'Essonne, France, 1999 (in French).

[4] G. Barish and K. Obraczka, World Wide Web Caching: Trends and Technologies, *IEEE Communications Magazine* 38/5 (2000) 178-184.

[5] F. Bellifemine and G. Rimassa, JADE – a FIPA-compliant Agent Framework. In: Proceedings of the Forth International Conference on the Practical Application of Intelligent Agents and Multi-Agents, London, 1999, 97-108.

[6] BoostWeb Overview: BoostWeb Optimizer v4., BoostWorks White Paper, Oct. 2001

[7] V. Cardellini, M. Colajanni and Ph.S. Yu, Dynamic Load Balancing on Web server, IEEE Internet Computing 3/3 (1999) 28-39.

[8] V. Cardellini, M.Colajanni and Ph.S. Yu, Geographic Load Balancing for Scalable Distributed Web Systems. In: Proceedings of the 8th International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, San Francisco, Aug./Sept. 2000.

[9] J. Challenger, A. Iyengar, P. Dantzig, D. Dias and N. Mills, Engineering Higly Accessed Web Sites for Performance. In: Web Engineering: Managing Diversity and Complexity of Web Application Development, LNCS 2016, Springer-Verlag, 2001, 247-265.

[10] J. Ferber, Multi-Agent Systems – An Introduction to Distributed Artificial Intelligence. Addison-Wesley, 1999.

[11] R. Fielding, J. Gettys, J.C. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, Hypertext Transfer Protocol – HTTP/1.1, RFC 2616, June 1999.

[12] Foundation for Intelligent Physical Agents. FIPA 97 specification: Agent Communication Language, Part 2, Version 2.0, 1997.

[13] Foundation for Intelligent Physical Agents. FIPA ACL Message Structure Specification, 2000.

[14] Java Agent Development Framework. http://sharon.cselt.it/projects/jade

[15] Y. Labrou, Y., Finin, T. History, State of the Art and Challenges for Agent Communication Languages, *Informatik – Informatique* 1 (2000) 17-24.

[16] J.-C. Mignot, Cache Web: un état de l'art des techniques et prototypes, *Technique et science informatique* 20/6 (2001) 719-748.

[17] A. Ramonaite, A. Contribution à la modélisation de systèmes multi-agents d'une architecture communicante pour l'accélération de flux applicatifs au travers de l'Internet. Tech. Report. Univ. Of Evry Val d'Essonne, France, Nov. 2000.