An agent-oriented architecture for peer-to-peer information integration system

Tai NGUYEN, Manuel KOLP
Université catholique de Louvain, IAG/ISYS – Information Systems Unit
{nguyen, kolp}@isys.ucl.ac.be

Abstract—Peer-to-peer (P2P) computing facilitates the sharing of computer resources and services by direct exchange between systems. In other hand, nowadays, the information integration systems are forced to build up open systems able to cope with distributed, heterogeneous, and dynamic information issues. Employing P2P can potentially provide a more open integration that invokes all participants on the network. These Multi-Agent Systems (MAS) architectures are gaining popularity for building open, distributed, and evolving software required by such systems. They tend to be open and dynamic in the sense they exist in a changing organizational and operational environment where new components can be added, modified or removed at any time like the heterogeneous sources on a P2P network. This paper presents an agent-oriented P2P architecture, based on the Gnutella protocol, specified for integrating information. In our system, we have applied a multi-criteria search engine to increase the efficiency in both aspects: the P2P network's search and the integration of information.

Index Terms—Multi-Agent Systems, organizational styles, peer-to-peer, information integration.

I. Introduction

Peer-to-peer (P2P) computing facilitates the sharing of computer resources and services by direct exchange between systems [Kan01]. Users can take advantage of existing computing power and networking connectivity to benefit their works. This architecture has achieved considerable attention by mainstream computer users and by the PC industry. The Napster [Shi01], MP3 music file sharing application, was set up in September 1999, and attracted more than 13.6 million users in February 2001 (source: comScore Media Metrix). If the Napster centralizes connections peer-peer by passing all interactions through a main server, the Gnutella [Kan01], invented in 1999, allows a client to connect to a server just one time and to get in touch with the set of clients of the network. Built on a GPL license, Gnutella is quickly developed within many projects like Beashare [Bea05], Limewire [LLC03], XoloX [Xol05] …

This paper presents an agent-oriented P2P architecture, based on the Gnutella protocol, specified for integrating information. Architectures for integrating information extracted from multiple heterogeneous sources allow exploiting effectively the numerous sources available on-line through the World Wide Web. Such architectures permit users to access and query numerous information sources to obtain an integrated answer. The sources may be conventional databases or other types of information, such as collections of Web pages. Employing P2P can potentially provide a more open integration that invokes all participants on the network. This leads to a significant increase in the efficiency.

Designing such information integration systems can rapidly become complex. Indeed, such processes require software architecture to operate within distributed environments that must evolve over time to cope with the dynamics and heterogeneity of information sources also with the dynamic set of participants. One promising source of ideas that has been considered in recent years for designing such information integration software is the area of Multi-Agent System (MAS) architectures. They appear to be more flexible, modular and robust than traditional including object-oriented ones. They tend to be open and dynamic in the sense they exist in a changing organizational and operational environment where new components can be added, modified or removed at any time like the heterogeneous sources on a P2P network.

In many P2P networks such as Napster and Gnutella's projects, the search is implemented by filename matching against a simple search string [MEA02]. In the information integration context, this limitation is more critical. In our system, we apply a multi-criteria search engine to increase the efficiency in both aspects: the P2P network's search and the integration of information. The search engine has been presented in [FKN+04]. This paper continues and integrates this research: it focuses on a multi-agent perspective for designing and specifying a P2P information integration architecture based on organizational styles. Organizational styles are instantiated to describe the roles of the agents in a specific architecture along with their properties and dependencies. In our case, the joint-venture organizational style will be instantiated to design the architecture of the system.

The paper is organized as follows. Section 2 introduces some perspectives of MAS with the BDI model and organizational styles. Section 3 describes briefly Gnutella protocol. Section 4 is reserved for our multi-agent approach on P2P information integration system development, including the design of the global architecture with organizational styles and

the corresponding implementation on an agent-oriented platform. Finally, Section 5 concludes the research.

II. MAS Architectures and Organizational Styles

A. The BDI Agent Models

An *agent* is defined as a system entity, situated in some environment that is capable of flexible autonomous action in order to meet its design objective [WJ96]. An agent can be useful as a stand-alone entity that delegates particular tasks on behalf of a user. However, in the overwhelming majority of cases, agents exist in an environment that contains other agents. Such environment is a *multi-agent system* that can be defined as a social and distributed organization of intelligent and autonomous software components that behave and collaborate to fulfill the objectives of an organizational structure [ZJW00].

In order to reason themselves and act in an autonomous way, many models have been used for define agents. They are usually built on rationale models and reasoning strategies that have roots in various disciplines including artificial intelligence, cognitive science, psychology and philosophy. Especially, the Belief-Desire-Intention (BDI) model [Bra88] has received a great deal of attention in studying the design rationale of agents because of its simplicity and intuitiveness with respect to human-based thinking [Fau04]. It has also been proposed as a keystone model in numerous agent-oriented development environments such as JACK [BRH+99], JADEX [Jad04] and the like.

The main concepts of the BDI agent model are in addition to the notion of agent itself we have just explained:

- The *beliefs* of an agent represent the current state of the environment as perceived by the agent
- The *desires* represent motivations of agents. That is what the agents is trying to achieve
- The *intentions* are selected desires to which an agent commits itself. The set of intentions is consistent and changes over time: the agent drops intentions that have already been achieved or that the agent believes to be unachievable, and the agent adopts new intentions (i.e., selects other desires to commit to).

B. Organizational Styles

The global organizational behavior of the MAS derives from the interaction among the constituent agents. To design such system, organizational styles are instantiated to describe the roles of the agents in a specific architecture along with their properties and dependencies. These styles are socially-based design alternatives inspired from models and concepts in organizational theories that analyze the structure, the design of real-world human organizations and the strategic cooperation between them [MSB99]. The advantages of using organizational styles in the process of MAS design are [Fau04]:

- Reducing the conceptual distance between the software system and the enterprise in which it has to operate;
- We, as social human beings, are continuously immersed in a variety of organizational contexts, which makes an organizational approach a natural way to perceive and organize the solution to a complex problem;
- The large amount of research in the area of organization and management theory has provided a substantial background of knowledge and experience that can be reused in the design of MASs.

We present here the joint-venture organizational style [DG99] which is used to design the GnuISYS system. In a few words, the joint-venture organizational style is a meta-structure that defines an organizational system that involves agreement between two or more partners to obtain mutual advantages (greater scale, a partial investment and to lower maintenance costs…). A common actor, the *joint manager*, assumes two roles: a *private interface role* to coordinate partners of the alliance, and a *public interface role* to take strategic decisions, define policy for the private interface, represent the interests of the whole partnership with respect to external stakeholders and ensure communication with the external actors. Each partner can control itself on a local dimension and interact directly with others to exchange resources, data and knowledge.

III. The Gnutella Protocol 0.4

We have applied the Gnutella protocol 0.4 in our system. We present briefly the description of this version. For more detail, please consult [Gnu05].

A. Servent

In Gnutella network, a client is also a server, and vice versa. These so-called Gnutella *servents* (or servents for short) perform tasks normally associated with both clients and servers. They provide client-side interfaces through which users can issue queries and view search results, while at the same time they also accept queries from other servents, check for matches against their local data set, and respond with applicable results. Due to its distributed nature, a network of Gnutella servents is highly fault-tolerant, as operation of the network will not be interrupted if a subset of servents goes offline [DNB03].

B. Description of the Gnutella protocol 0.4

The Gnutella protocol defines the way in which servents communicate over the network. It consists of a set of descriptors used for communicating data between servents and a set of rules governing the inter-servent exchange of descriptors. Currently, there are five defined descriptors: *Ping*, *Pong*, *Query*, *QueryHit* and *Push*

- *Ping*: Used to actively discover hosts on the network. A servent receiving a Ping descriptor is expected to respond with one or more Pong descriptors.
- *Pong*: The response to a Ping. Includes the address of a connected Gnutella servent and information regarding the amount of data it is making available to the network
- *Query:* The primary mechanism for searching the distributed network. A servent receiving a Query descriptor will respond with a QueryHit if a match is found against its local data set.

- *QueryHit:* The response to a Query. This descriptor provides the recipient with enough information to acquire the data matching the corresponding Query.
- *Push:* A mechanism that allows a firewalled servent to contribute file-based data to the network

1) Descriptor Routing: The peer-to-peer nature of the Gnutella network requires servents to route network traffic (queries, query replies, push requests, etc.) appropriately. There are rules for descriptors routing. These routing are presented in the Figure 1 and 2. A well-behaved Gnutella servent will route protocol descriptors according to the following rules:

1. Pong descriptors may only be sent along the same path that carried the incoming Ping descriptor. This ensures that only those servents that routed the Ping descriptor will see the Pong descriptor in response. A servent that receives a Pong descriptor with Descriptor ID = n, but has not seen a Ping descriptor with Descriptor ID = n should remove the Pong descriptor from the network.

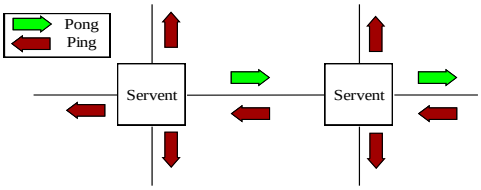

Figure 1 : Ping/Pong Routing

2. QueryHit descriptors may only be sent along the same path that carried the incoming Query descriptor. This ensures that only those servents that routed the Query descriptor will see the QueryHit descriptor in response. A servent that receives a QueryHit descriptor with Descriptor ID = n, but has not seen a Query descriptor with Descriptor ID = n should remove the QueryHit descriptor from the network.
3. Push descriptors may only be sent along the same path that carried the incoming QueryHit descriptor. This ensures that only those servents that routed the QueryHit descriptor will see the Push descriptor. A servent that receives a Push descriptor with Descriptor ID = n, but has not seen a QueryHit descriptor with Descriptor ID = n should remove the Push descriptor from the network.

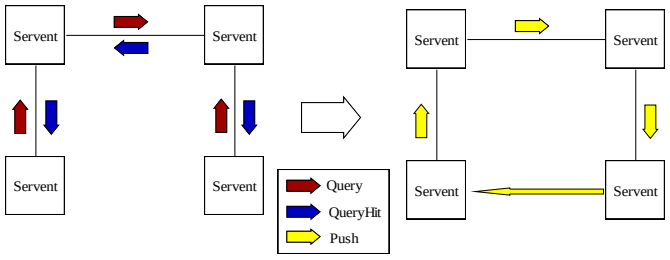

Figure 2 : Query/QueryHit/Push Routing

4. A servent will forward incoming Ping and Query descriptors to all of its directly connected servents, except the one that delivered the incoming Ping or Query.
5. A servent will decrement a descriptor header's TTL field, and increment its Hops field, before it forwards the descriptor to any directly connected servent. If, after decrementing the header's TTL field, the TTL field is found to be zero, the descriptor is not forwarded along any connection.
6. A servent receiving a descriptor with the same Payload Descriptor and Descriptor ID as one it has received before, should attempt to avoid forwarding the descriptor to any connected servent. Its intended recipients have already received such a descriptor, and sending it again merely wastes network bandwidth.

2) File Downloads

Once a servent receives a QueryHit descriptor, it may initiate the direct download of one of the files described by the descriptor's Result Set. Files are downloaded out-of-network i.e. a direct connection between the source and target servent is established in order to perform the data transfer. File data is never transferred over the Gnutella network.

The Gnutella protocol provides support for the HTTP Range parameter, so that interrupted downloads may be resumed at the point at which they terminated.

IV. MAS Architecture for P2P Information Integration

GnuISYS is a typical P2P information integration application we have developed using the concepts explained in Section 2 and 3. The application provides a Multi-Agent System architecture to support the P2P integration of information with different heterogeneous sources

1) GnuISYS Network Architecture

The GnuISYS developed can be divided into four major components

- *The GUI-Connection:* It is used to link the end user with the application and between the application's components.
- *Gnutella Protocol:* this is the backbone of the application. The Gnutella Protocol is used to communicate the peers on the network.
- *Search Engine:* The Search Engine is responsible for searching and integrating information from different heterogeneous sources.
- *Multi-criteria Analyze:* The Multi-criteria Analyze is used to express the user preferences in a more detailed way, and refine the possible domain of results

The collaboration of these components is shown in the Figure 3. At a peer, an end-user sends query to the GnuISYS. The Multicriteria Analyze and The Search Engine (Query Dispatch) will collaborate to rewrite the query into set of sub-queries. The Search Engine (Local Search) search the local resources for each sub-query. Each of sub-queries will also be send to another peer for more results. Each of sub-results, coming from local Search Engine or from the network, will be passed to the Multicriteria Analyze to evaluate its pertinence and then send to the Search Engine (Results Merge) to build up the temporary final result. At any moment a sub result is send back to Results Merge at the user's peer, the temporary final result will be re-arranged. The temporary final

result becomes permanent if and only if the propagation of sub-queries is stopped. The propagation of sub-queries can be stopped by the user (if he does not wish to continue this search) or by the network (if message's time-to-live is zero).

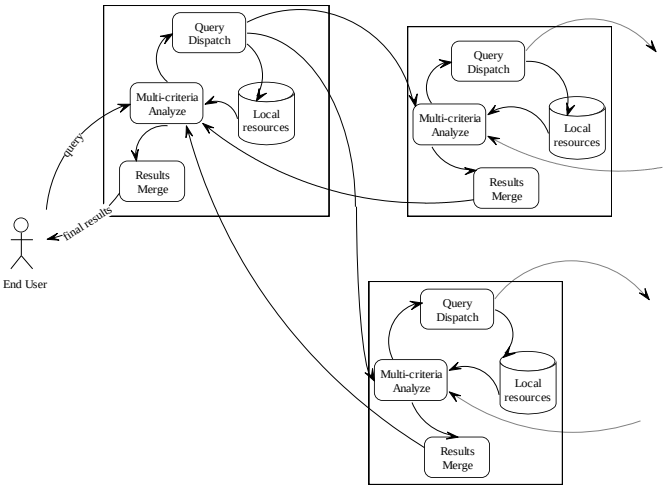

Figure 3 : The components of GnuISYS

2) GnuISYS Organizational Architecture

Figure 4 models the organizational architecture of GnuISYS using the i* model [Yu95] following the joint-venture organizational style we have introduced in Section 2. Briefly, i* is a graph, where each node represents an actor (or system component) and each link between two actors indicates that one actor depends on the other for some goal to be attained. A dependency describes an "agreement" (called *dependum*) between two actors: the *depender* and the *dependee*. The depender is the depending actor, and the dependee, the actor who is depended upon. The type of the dependency describes the nature of the agreement. Goal dependencies represent delegation of responsibility for fulfilling a goal; softgoal dependencies are similar to goal dependencies, but their fulfillment cannot be defined precisely; task dependencies are used in situations where the dependee is required. As show in Figure 4, actors are represented as circles; dependums – goals, softgoals, tasks and resources – are respectively represented as ovals, clouds, hexagons and rectangles; dependencies have the form depender → dependum → dependee.

Figure 4 shows that the Servent plays the role of the joint manager private interface, other joint venture partners are the Upload Manager, Download Manager, Search Engine, Multicriteria Analyze and File Lister. The public interface is assumed by the Connection Management.

The agent Connection Management is responsible for connecting the application to the network. It manages messages of the descriptive type coming from the network.

The agent Servent is the interface between the application and the network. All messages coming from the network or going out from the application must pass through this agent.

The agent File Lister shows files in a shared directory to requesters. If a change occurs in the directory, it will announce this change to all requesters who have asked for its list of files.

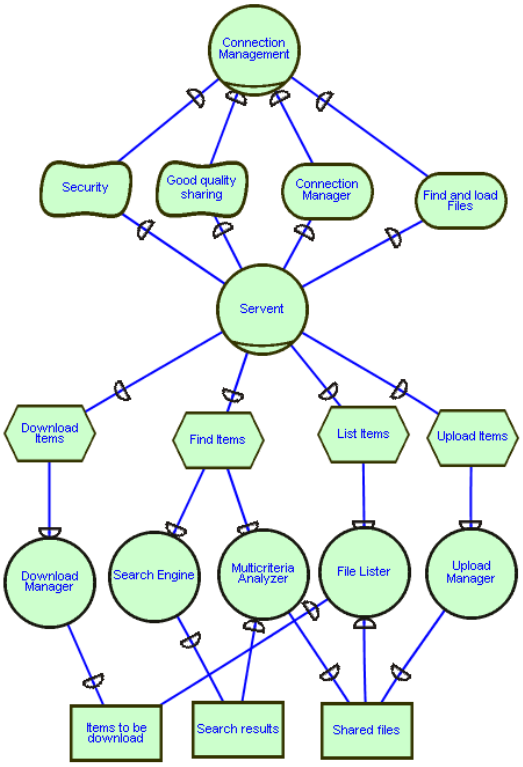

Figure 4 : The GnuISYS Architecture in joint-venture

The Upload(Download) Manager is responsible for uploading (downloading) files and confirms to File Lister the files which are being uploaded (downloaded).

In this system, the Search Engine is responsible for searching and integrating information from different local heterogeneous sources. Local sources can be on the peer's hard disks or peer's LAN network. The integrated result is shown in a text file. By sharing this file, we can share the integration of information. An integration of these shared text files presents a global view of the information on the whole network.

To extend the search efficiency of P2P network, we have developed a Multi-criteria Analyze in order to express the user preferences in a more detailed way, and refine the possible domain of results. We have applied method lexicography [Vin97] to sort items from many sub-results by taking care of user's preferences. By this way, we can build a final result in which each item will be evaluated by its pertinence.

2) GnuISYS at work

GnuISYS starts by connecting to the network. At the beginning, GnuISYS shows the interface, illustrated in Figure 5, with the information on the network such as: connected servers, send and received messages and a graphical table of various speeds of connections according to time. The status bar of the program is at the bottom of interface. It shows important information of the program. The first zone, from the

left, enables us to know immediately if the application is connected to one or several servents. If no connection is established, then the icon will be red and one will be able to read there the message "Connection down". On the other hand, if there is at least one established connection, the message will be "Connection up" and a green icon will take the place of the red. The second zone is the status of this servent. Like the status of connection, the servent can be "Up" or "Down". The third zone presents the number and the size of shared files. The last zone on the right is presented for a direct visualization of upload/download speeds.

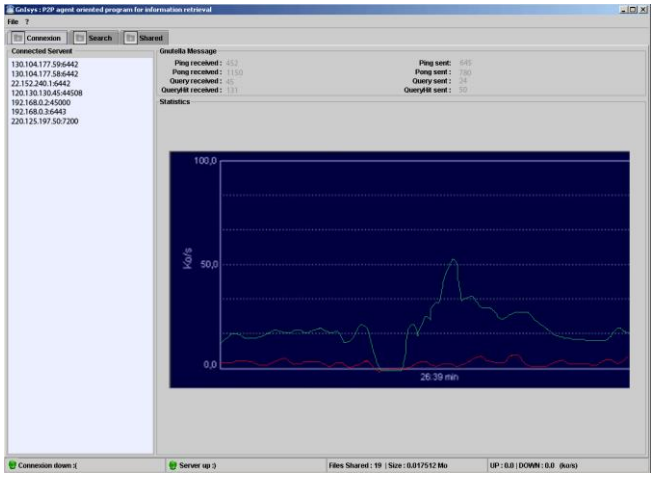

Figure 5: Interface of Connection Management

The Figure 6 composes the information coming from three agents: Upload Management, Download Management and File Lister. The agent Upload/Download Management shows temporary files, which are being uploaded or downloaded. The agent File Lister shows information of downloaded files in shared directory. The right side is reserved for File Info Display, which can play a video or music file (not implemented yet), show the content of a text file… (as illustrated in Figure 6)

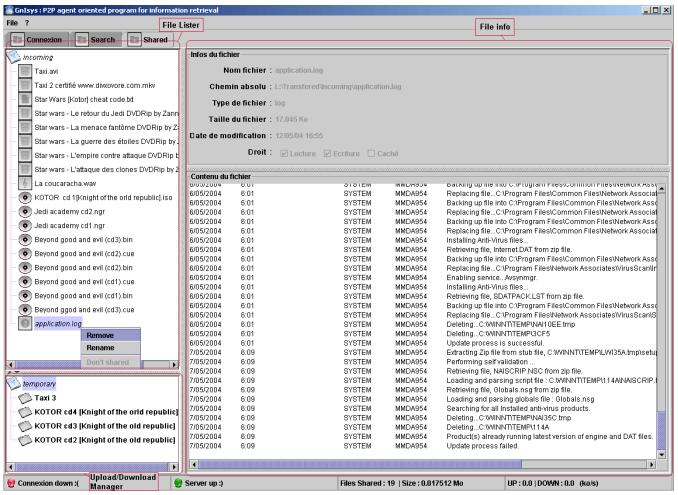

Figure 6: Shared Files

The Figure 7 presents the search function of GnuISYS. To launch a search, the user gives the keywords and determines the criteria.

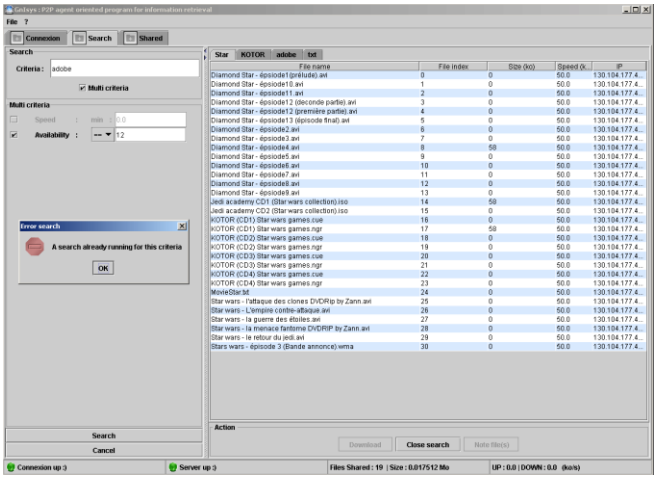

Figure 7: Search files

The user can do many searches at the same time. Each search is executed continuously until the user wishes to stop finding files or the messages' time-to-live is zero.

V. CONCLUSION

Nowadays, the information integration systems are forced to build up open systems able to cope with distributed, heterogeneous, and dynamic information issues. Most of these software systems exist in a changing organizational and operational environment where new components can be added, modified or removed at any time. In addition, the current success of P2P file sharing applications has highlighted the benefits of distribution and redundancy of resources [MEA02]. Together, the two paradigms can provide the essential infrastructure for executing dynamic integration over the Internet. Such integration is adaptive (controlled by the peers) and easily scaleable (i.e. by altering the number of participating peers) [Fla01]. This paper has described a multi-agent systems architecture based on organizational styles to design a P2P information integration system. The paper has proposed a validation of the architecture: it has been applied to develop GnuISYS, a Gnutella-based P2P information integration platform implemented on the JACK agent development environment.

REFERENCES

[Bea05] BearShare, http://www.bearshare.com, 2005

[Bra88] M Bratman. Intentions, Plans and Practical Reasoning. Havard University Press, 1988

[BRH+99] P. Busetta, R. Ronnquist, A. Hodgson and A. Lucas *JACK intelligent agents – components for intelligent agents in java* AgentLink News Letter, January 1999. White paper, http://www.agent-software.com.au

[DG99] P. Dussauge and B. Garrette. *Cooperative Strategy: Competing Successfully Through Strategic Alliances*. Wiley and Sons, 1999.

[DKFP04] T. Tung Do, Manuel Kolp, Stéphane Faulkner, Alain Pirotte, *Agent-Oriented Design Patterns*, Proc. of the 6th Int. Conf. on Enterprise Information Systems ICEIS'04, Porto, Portugal, April 14-17, 2004, pp. 48-53

[DNB03] C. H. Ding, S. Nutanong, R. Buyya, "Peer-to-Peer Networks for Content Sharing", Technical Report, GRIDS-TR-2003-7, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, December 2003

[Fau04] Faulkner S. *An Architectural Framework for Describing BDI Multi-Agent Information Systems* Ph. D. Thesis, Université Catholique de Louvain, Institut d'Administration et de Gestion (IAG), Louvain-la-Neuve, Belgium, May 2004.

[FKN+04] Stéphane Faulkner, Manuel Kolp, Tai Nguyen, Adrien Coyette, Tung Do. *Information Integration Architecture Development: A Multi-Agent Approach*. Software Engineering and Knowledge Engineering (SEKE), 2004

[Fla01] G. Flammia, *Peer-to-Peer is not for everyone*, Internet Services, IEEE Intelligent Systems, 2001

[GKM03] Paolo Giorgini, Manuel Kolp, John Mylopoulos: *Multi-agent and Software Architectures: A Comparative Case Study*, In Agent-Oriented Software Engineering III, Third International Workshop, AOSE'02, Bologna, Italy, July 15, 2002, Revised Papers and Invited Contributions, 2003, pp.101—112.

[Gnu05] Gnutella Specification http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf , 2005

[Jad04] Project JADEX, http://sourceforge.net/projects/jadex, 2004

[Kan01] Kan, G., *Gnutella*, Peer-to-Peer:Harnessing the Power of Disruptive Technologies, a. Eram (ed.), O'Reilly Press, USA, 2001

[KGM02] Manuel Kolp, Paolo Giorgini and John Mylopoulos, *Organizational multi-agent architectures: a mobile robot example*, In Proc. of the 1st Int. Joint Conf. on Autonomous Agents & Multiagent Systems, AAMAS'02, July 15-19, 2002, Bologna, Italy, pp. 94-95.

[KM01] Manuel Kolp, John Mylopoulos, *Software architectures as organizational structures*, Proc. ASERC Workshop on" The Role of Software Architectures in the Construction, Evolution, and Reuse of Software Systems", Edmonton, Canada

[LLC03] LimeWire LLC, *Improving the Gnutella Network*, http://www.limewire.com/index.jsp/im_require, 2003

[MEA02] A. Mukherjee, B. Esfandiari, and N. Arthorne. *A Peer-to-peer System for Description and Discovery of Resource-sharing Communities.* In RESH, 2002.

[MKFG05] Haralambos Mouratidis, Manuel Kolp, Stephane Faulkner, and Paolo Giorgini, *A secure architectural description language for agent systems*. In Proc. of the 4th int. joint conf. on Autonomous agents and multiagent systems, AAMAS '05, July 25-29, Utrecht, The Netherlands, pp 578-585

[MKP02] John Mylopoulos, Manuel Kolp and Paolo Giorgini, *Agent-Oriented Software Development*, in Proc. of the 2nd Hellenic Conf. on AI, SETN'02, *Methods and Applications ofArtificial Intelligence*, Thessaloniki, Greece, April 11-12, 2002, pp. 3-17.

[MSB99] J. Morabito, I. Sack and A. Bhate. *Organization modeling : innovative architectures for the 21st century*, Upper Saddle River, N.J., Prentice Hall PTR, 1999.

[Shi01] Shirky, C., *Listening to Napster*, Peer-to-Peer: Harnessing the Power of Disruptive Technologies, A. Oram (ed.), O'Reilly Press, USA, 2001

[Vin97] P. Vincke. *L'aide multicritère à la décision*. Éditions de l'université de Bruxelles et Éditions Ellipses. 1997.

[WJ96] M. Wooldridge and N.R Jennings, editors. *Special Issue on Intelligent Agents and Multi-Agent Systems.* Applied Artificial Intelligence Journal. Vol. 9(4), 1996.

[Xol05] XoloX, http://www.xolox.nl, 2005

[Yu95] Yu E. *Modeling Strategic Relationships for Process Reengineering*, Ph.D. thesis, Department of Computer Science, University of Toronto, Canada, 1995.

[ZJW00] F. Zambonelli, N.R. Jennings and M. Wooldridge. Organizational abstractions for the analysis and design of multi-agent systems. In Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering, volume 1957 of LNCS, pages 243-252, Springer Verlag, 2000.