A simulator for a two layer MAS adaptation in P2P networks

Jordi Campos[1], Marc Esteva[2], Maite López-Sánchez[1] and Javier Morales[2]

[1] MAiA Deptartment, Universitat de Barcelona, email: {jcampos,maite}@maia.ub.es
[2] Artificial Intelligence Research Institute (IIIA), CSIC,
email:{marc,jmorales}@iiia.csic.es

Abstract. Adapting organisational structures to maintain an organisation effectiveness under varying circumstances is becoming a hot topic within the agent community. In this paper we present a simulator that we have developed for testing adaptation mechanisms in Peer to Peer scenarios. We regard this service as part of the new generation of services that should be incorporated into multiagent systems infrastructures to assist coordination both at participant and organisation levels. In order to provide such organisational adaptation we rely on an added distributed meta-level. Meta-level agents perceive partial information about system properties that they use to adapt organizational structures when necessary. The simulator implements different sharing methods, allows to define different network topologies, and includes some facilities to process and analyse simulation results in order to compare them.

1 Introduction

Organisational structures have proven to be useful to regulate MultiAgent Systems (MAS) [1, 2]. However, certain environmental or population changes may imply a decrease in goal fulfilment. Thus, adapting organisations is now becoming an active research area [3–6], since it can help to keep the expected system outcomes under changing circumstances.

In particular, we propose to add a *meta-level* in charge of adapting system's organisation instead of expecting agents to increase their behaviour complexity. This is specially relevant when dealing with open MAS, since there is no insight of participant's implementation, and hence, we can not guarantee that agents are endowed with organisational adaptation capabilities. We regard this adaptation –together with other possible *meta-level* functionalities– as an assistance to agents that can be provided by MAS infrastructure. Thus, we call our implementation approach Two Level Assisted MAS Architecture (2-LAMA)[7]. Furthermore, in order to avoid centralisation limitations such as fault-tolerance or global information unavailability, we propose this architecture has a distributed *meta-level* —i.e., it is composed of several agents. In this context, in this paper we present a simulator for testing organisational adaptation mechanisms in P2P scenarios. Hence, the main goal of the paper is to describe the simulator functionality.

Our approach is able to deal both with highly dynamic environments and domains without direct mapping between goals and tasks —i.e. domains where it is not possible to derive a set of tasks that achieve certain goals. Nevertheless, it requires domains where organisations can be dynamically changed. Peer-to-Peer sharing networks (P2P) present all previous features, and thus, we use them as a case study. In such networks, computers contact among them to share some data and their relationships change over time depending on network status and participants. In this context, a P2P system is modelled as an organisation having a social structure among peers and a set of protocols and norms that regulate the sharing process. On top of the P2P system –that we call domain-level– we add a distributed meta-level that perceives status information and uses it to adapt peers' social structure and norm values. Meta-level adaptation is based on system performance, which is measured by the time peers spent to share data and the required network consumption.

This paper is structured as follows. Next section provides a more detailed description of the 2-LAMA model, so that Section 3 can apply it to the P2P scenario. This scenario has been implemented in the simulator presented in Section 4. Finally, some conclusions and future work are described in last section.

2 2-LAMA Model

Organisational structures regulate MAS by providing an agent coordination framework and some domain independent services that alleviate agent development. We regard these services as *Coordination Support* services [8] that include basic coordination elements such as elemental connectivity or agent communication languages. Usually, these services are devoted to enact agent coordination at different levels. In addition to them, we propose new services that provide an added value by assisting coordination further than just enabling it. In this manner, we group services in two different layers: an *Organisational Layer* that provides coordination enabling services, and an *Assistance Layer* on top of it, which provides coordination assistance services. This last layer includes a proactive service that adapts organisations depending on system's evolution.

The *Organisational Layer* provides basic services supporting the enactment of the organisation. We denote an organisation as $Org = \langle SocStr, SocConv, Goals \rangle$ where:

- *SocStr* corresponds to the social structure, which consists of a set of roles and the relationships among them.
- *SocConv* are social conventions that agents should conform and expect others to conform [9]. They are expressed as norms and/or interaction protocols, which define legitimate sequences of actions performed by agents playing certain roles.
- *Goals* describe the organisation design purpose —as opposed to agents' individual goals. They are expressed as a function on certain observable properties so that the system can evaluate its own performance.

Regarding our *Assistance Layer*, it provides two main types of services [8]: *Agent Assistance* and *Organisational Assistance*. The former assists individual agents to follow current social conventions. It includes four different services: *Information* that is useful for participating in the MAS; *Justification* of specific actions' consequences; *Advice* of alternative plans conforming social conventions; and *Estimation* of action consequences due to current conventions. The latter, the *Organisational Assistance*, consists on adapting the existing organisation in order to improve system's performance under varying circumstances. To provide such an adaptation, we propose goal fulfilment as its driving force within the context of a rational world assumption. Hence, the *Assistance Layer* requires some way (i) to observe system evolution, (ii) to compare it with the organisational goals and (iii) to adapt the organisation trying to improve goal fulfilment.

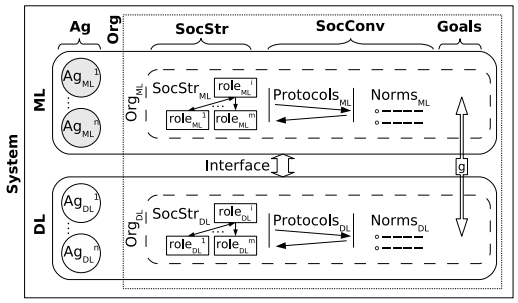

Fig. 1. Two Level Assisted MAS Architecture(2-LAMA): Domain Level (DL), Meta Level (ML) and Interface.

In order to provide *Assistance Layer*'s services, we have proposed a *Two Level Assisted MAS Architecture* (2-LAMA, [7]). It consists on a distributed *meta-level* (ML) that provides assistance to a *domain-level* (DL) in charge of domain-specific tasks. Figure 1 shows them and their communication trough an interface (Int). Thus, the whole system can be expressed as $2LAMA = \langle ML, DL, Int \rangle$[3]. Each level has a set of agents with its own organisation: $DL = \langle Ag_{DL}, Org_{DL} \rangle$ and $ML = \langle Ag_{ML}, Org_{ML} \rangle$. Using the interface, ML agents perceive partial information[4] about environmental observable properties ($EnvP$, e.g. date or temperature) and agents' observable properties (AgP, e.g. colour or position). In particular, a ML agent has partial information about the subset of DL agents it assists. We assume DL agents are grouped into clusters according to a domain-specific criterion —e.g. interaction costs. Therefore, a ML agent –we call it *assistant*– assists a cluster of DL agents, observes partial information about them, and shares it with other ML agents in order to provide better assistance services.

[3] It is possible to nest subsequent *meta-levels* updating previous level's organisation.
[4] In many scenarios global information is not available.

3 P2P model

Our case study is a simplified version of real Peer-to-Peer sharing networks (P2P), where a set of computers connected to the Internet (peers) share some data. This setting represents a highly dynamic and complex scenario, and thus, it is suitable for the development of a simulator implementing our approach.

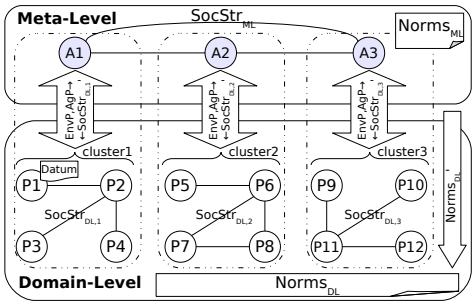

Fig. 2. 2-LAMA in the P2P scenario. Agents: peers P1..P12 at DL and assistants A1..A3 at ML.

Figure 2 shows our P2P model, where the DL is composed by agents playing the peer role. DL *social structure* determines agents' relationships, which corresponds to the neighours peers contact to share the data. The organisational goal (*Goals*) is that all peers receive the data with the minimal time and network consumptions. Hence, the time needed to share the data and the network consumption are the two metrics to evaluate simulation results. The *social conventions* at DL include the sharing protocol specified below and two norms. First *norm* limits agents' network usage in percentage of its nominal *bandwidth*.This *norm* can be expressed as: $normBW_{DL}$ ="a peer cannot use more than $\texttt{max}_{\texttt{BW}}$ *bandwidth* percentage to share data". This way, it prevents peers from massively using their *bandwidth* to send/receive data to/from all other peers. Second *norm* limits the number of peers to whom a peer can simultaneously send the data. Hence, we define $normFriends_{DL}$ ="a peer cannot simultaneously send the data to more than $\texttt{max}_{\texttt{Friends}}$ peers".

As we have already mentioned, ML provides assistance to DL. Each ML agent plays the *assistant* role for a cluster of DL agents (peers). It does it so by collecting information and adapting their local organisation. Its decisions are based on local information about its cluster, aggregated information about other clusters and the norms at ML. Some examples of local information are latencies ($EnvP$) or which peers have the data (AgP). Information about other clusters come from other *assistants* in the ML *social structure*. As for ML norms, we consider one limiting the number of peers –in the *cluster*– to inform about a new peer –in another cluster– having the data. Thus, we define $normHas_{ML}$ ="Upon reception of a completed peer (p $\notin$ *cluster*) message, inform no more than $\texttt{max}_{\texttt{Has}}$

Phase	Level	Protocol Messages
initial	ML	join<hasDatum>
latency	ML	get_latency<peers>, latency<peer><measure>
	DL	lat_req, lat_rpl
social structure	ML	contact<peers>
handshake	DL	bitfield<hasDatum>
data sharing	DL	request, data, cancel
	ML	completed, completed_peer<peer>, has_datum<peer>, all_completed
inactive	DL	have
waiting	DL	choke, unchoke
norms	ML	suggested_bw<value>, suggested_friends<value>, norm_updated<norm_id><new_definition>

Table 1. Protocol messages grouped into subsequent phases.

peers $\in$ *cluster* ". Finally, we assume *assistants* are located at Internet Service Providers (ISP), and thus, related communications are fast.

3.1 Protocol

Our proposed protocol is a simplified version of the widely used BitTorrent [10] protocol[5]. Table 1 presents the messages that are exchanged during protocol phases. Notice that the table includes the messages among DL agents, but also messages involving assistants at ML. Initially, peers join their cluster by informing its assistant. Afterwards, in order to compute the *social structure*, assistants need local information and therefore, they initiate latency phases requesting peers to measure their latency with all other peers in their clusters. Assistants use this information to propose a social structure among peers in their clusters. The social structure defines the overlay network within a cluster —i.e. which peers each peer has to contact in order to obtain the data.

Thereafter, peers perform a handshake phase where they introduce themselves to their contacts, and specify whether they have the datum. If this is the case, a data sharing phase starts —including data request and data transmission. Otherwise, as soon as one peer receives the datum, it will inform its handshaked peers so that the sharing phase is triggered this time. Nevertheless, upon request, a source peer cannot start a transmission if it is already serving the maximum number of allowed peers (defined by the $\texttt{max}_{\texttt{Friends}}$ value). For those cases, transmission can only be initiated when a previous transmission ends.

Upon data reception, a peer also informs its assistant. Then, this assistant shares this information with other assistants, who, in turn, inform some ($\texttt{max}_{\texttt{Has}}$) peers, so new data sharing phases can be started. An assistant also informs other assistants when all peers in its cluster are completed preventing further unnecessary communications. Finally, norm deliberations and notifications also belong to the protocol.

[5] Specially, we assume the information is composed of a single piece of data.

4 Simulator

As a platform to run our experiments, we have implemented a P2P sharing network simulator in Repast Simphony. Its architecture allows to both model agents (*agent-level*) and the transport of messages among them (*network-level*). The model that simulates the message transport is a packet switching network. Simulation at network level allows us to compare different P2P approaches taking into account the environmental changes that occur at this level (notice that this feature is not present in Repast Simphony simulator framework). In our current implementation, network status just depends on MAS activity, but we could introduce additional traffic that disrupts it. Our simulator includes our 2-LAMA approach and the standard BitTorrent protocol, and provides some facilities to collect information at agent/network level and to analyse it.

4.1 Architecture

Our simulator has an internal architecture that clearly isolates different functionalities. On the one hand, we have a module called `p2p` that represents the conceptual model defined by the 2-LAMA and is targeted to drive the simulation at *agent-level*. It provides tools to create state-based agents, to define a problem (number of peers, who has initially the datum, etc.) or services such as an agents' directory. The upper part of figure 3 shows the P2P implementation of the 2-LAMA architecture described in previous section.

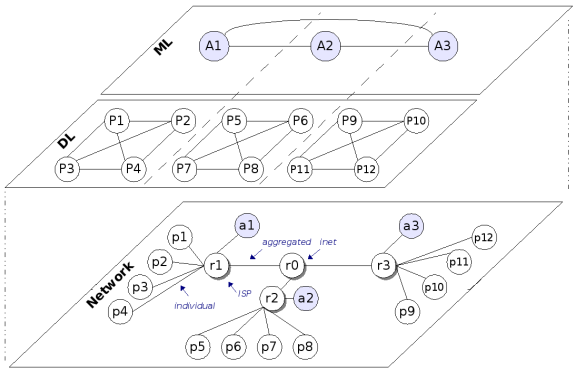

Fig. 3. 2-LAMA model and the underlying network. Agents are modelled on the top but their exchanged messages traverse the network on the bottom.

On the other hand, we have a module that drives the simulation at network-level. This module is called `netsim` and provides facilities to transport messages among agents, to define different network topologies, and to collect statistical

information about network status. In order to use this module, an agent from the `p2p` module is attached to a `netsim`'s network adapter, which is in charge of actually sending messages. These messages are split into packets that travel along links and follow their path by switching at routers. The destination agent is informed when each packet reaches its network adapter. Eventually, when all packets of a message arrive, the network adapter also delivers the whole message to the destination agent. Hence, agents can pay attention to packets or just wait for entire messages. The latency of a message from one network adapter to another depends on the number of links, their *bandwidth* and the current traffic through them.

The lower part of figure 3 depicts the `netsim` module, which exemplifies a network topology. We can see how peers with a good communication among them are grouped into a cluster and have individual links to the same ISP. In this example, peers `P1` and `P2` are connected at conceptual level, which at network level is achieved by connecting their corresponding network terminations `p1` and `p2` to the same ISP (`r1`). We also have the agent `A1`, which is the assistant of these peers, connected to the same ISP (`r1`) through its corresponding network termination `a1`. Each cluster is connected to the others by means of links, so `r1` and `r2` have aggregated links connected to `r0`, which represents the interconnection through Internet.

The simulator also includes an `overepast` module that processes the generated logs and extracts relevant information. This information can be later on displayed in different types of graphics. Hence, this can be used to compare the time spent to share the data in different configurations, or by using different sharing methods.

4.2 Sharing methods

Our simulator offers alternate sharing methods so that they can be executed over the same initial configurations in order to compare their results. Current available methods are: a single-piece version of the BitTorrent protocol (BT), the 2-LAMA approach with *social structure* adaptation but no *norm* adaptation and the 2-LAMA approach with *social structure* and *norm* adaptation.

Since the BitTorrent protocol inspired our 2-LAMA P2P approach, both protocols at peers' level are really similar (in fact, both protocols work with single-piece data and share most messages). Latency phase is not present in BT and our whole ML collapses into a single agent (*tracker*) that informs about all existing peers. As a consequence, the social structure phase is reduced to the tracker informing about all connected peers, ant thus, peers do not receive any further assistance to share the datum. Data sharing phase follows the algorithm described in [10]. In brief, it uses the same messages but decisions do not depend on norms but on protocol pre-fixed variables, so agents do not have any chance to take their own decisions.

In contrast, agents in the 2-LAMA approach can decide their actions as far as they respect norms. When executing the 2-LAMA approach *social structure* adaptation but no *norm* adaptation, *norm* parameters ($\texttt{max}_{\texttt{BW}}$, $\texttt{max}_{\texttt{Friends}}$, $\texttt{max}_{\texttt{Has}}$)

are fixed from start. On the contrary, the 2-LAMA approach with *social structure* and *norm* adaptation also has the *norm* parameters, but $\texttt{max}_{\texttt{BW}}$ and $\texttt{max}_{\texttt{Friends}}$ are self-updated at run-time at certain adaptation intervals ($\texttt{adapt}_{\texttt{interv}}$, an additional parameter). Each assistant computes their desired values for each norm taking into account the information collected from its cluster and the information received from other assistants. Assistants use a voting scheme as a group decision mechanism to choose the actual norm updates before notifying their peers.

4.3 Graphical User Interface (GUI)

By its very nature, MAS are complex systems composed by many agents acting and interacting simultaneously. Runtime monitoring information is usually low level, so we need graphical means to analyse system's evolution from a higher level of abstraction. With this aim, our simulator extends the Repast GUI and creates and advanced user-friendly GUI. Next sections detail its architecture and the new functionalities it provides.

Architecture Figure 4 shows the two main parts of the simulator: the core and the GUI. The core is based in Repast Simphony, which provides general simulation utilities such as schedulers or basic extendable models. By extending it, we have created our 2-LAMA P2P Simulator core, which implements domain-specific simulations. As for the GUI, our advanced GUI also extends Repast original GUI, providing some additional functionalities. It is able to exchange information with it and uses its basic tools to draw elements in the screen, such as the agents in the simulation, the messages sent among them, etc.

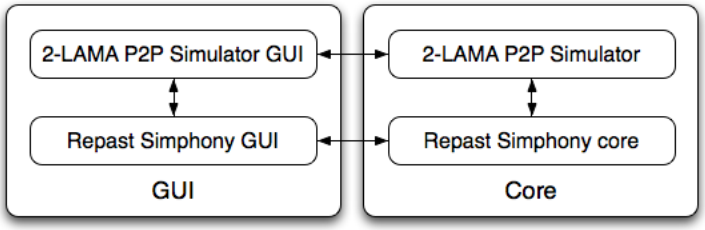

Fig. 4. Repast-based architecture of the 2-LAMA Simulator GUI

Our GUI also captures information from the P2P simulator in order to obtain all the information that will be displayed subsequently. This information comes down to the messages that are being sent among peers. The GUI is constantly listening to the simulator to catch these messages, which are stored afterwards. Figure 5 depicts the organisation of these messages. Peers are grouped in pairs, and these pairs are labelled with the name of its peers alphabetically ordered. Each pair of peers has a bag with the messages that one peer is sending to the

other at a given time step. For example, left side of the figure shows a group labelled `P1P2`. This group has a bag storing three messages from peers `P2` and `P1`, and the first one is a `PIECE` message that is being sent from `P2` to `P1`. The GUI uses this information to paint coloured arrows that represent the type and direction of messages that are being sent among peers.

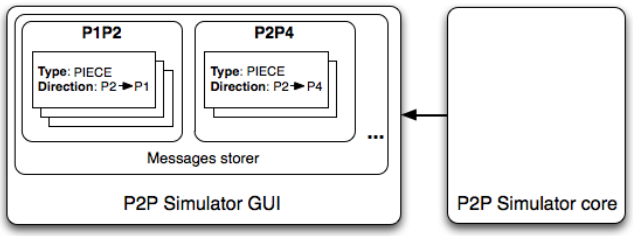

Fig. 5. Organisation of the information needed by the GUI to display simulation events

Functionalities This section explains the functionalities provided by our advanced GUI. Figure 6 depicts a screenshot of our simulator that illustrates GUI's general appearance. As for any other GUI, it has the natural aim of supporting user's interaction and the general objective of providing relevant information about the simulation —such as the messages sent at a given time step, their type, the source and target peers of the message, etc.

As Figure 6 shows, GUI functionalities are distributed in the following six main layout areas:

1. **Control toolbar**: This toolbar pertains to the original Repast GUI (and thus, it does not requires further implementation). It allows users to play the simulation, pause it or execute it step by step, where each step corresponds to a *tick* of the simulation.
2. **Legend of agents**: It shows how the different types of agents and possible states are displayed in the layout. Thus, the user can identify each agent and know if it is acting as an assistant or a peer, and have it into account to interpret what is happening in the simulation at every moment.
3. **Legend of message types**: It shows the colours corresponding to each type of message agents can exchange. This colour can be changed by means of the element on the left of the coloured line next to each message type. These changes can be saved into a file to recover them in future simulations.
4. ***Visible* and *Pause* checkboxes**: For each type of message, there are also two checkboxes that allow the user to show/hide the messages of that type that are exchanged among agents, or pause the simulation when any agent sends a message of that type.
5. **Runtime P2P Network layout**: This layout shows an animation of the agents of the simulation and the communications among them. Peers and

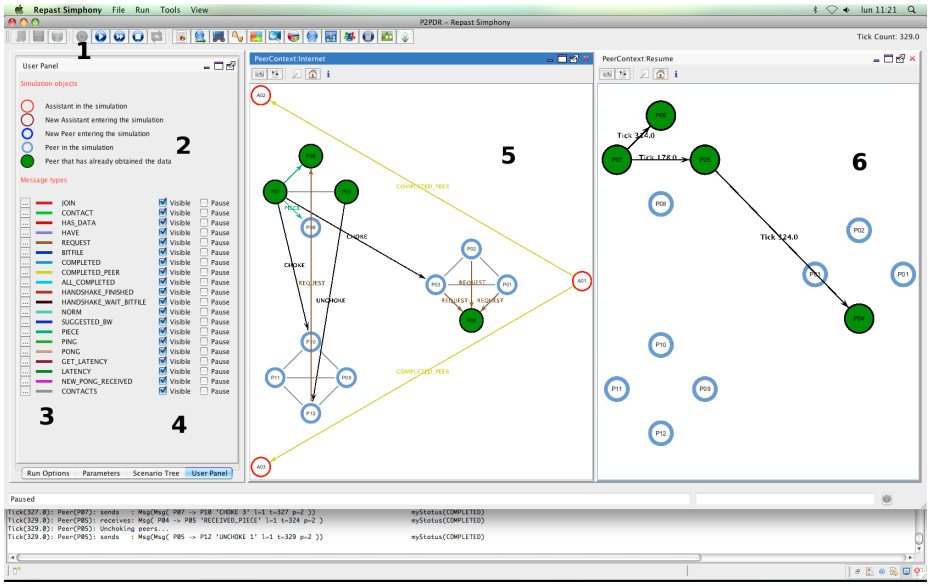

Fig. 6. 2-LAMA P2P Simulator Graphic User Interface

assistants are drawn according to the network topology. Following the example of Figure 6, peers are grouped in clusters of four peers, where each cluster is linked to an assistant. Messages sent among agents are displayed according to the defined colour in the user panel.

6. **Resume layout**: This layout shows how the data has been distributed among the P2P agent community. It also highlights completed peers and displays arrows connecting source and receiver agents. Furthermore, these arrows are labelled to specify at what time step the data was received.

Both *Runtime P2P Network* layout and *Resume* layout are able to draw new agents entering the simulation, highlighting them during some *ticks* and redistributing the layout to place them into the network.

4.4 Results analysis facilities

As it has been previously mentioned, our `overepast` module collects textual information (logs) and analyses it off-line (i.e., at the end of one or several executions). Analysis is done by generating additional plots that show how the main metrics change along execution time or with different simulation parameters. Thus, summarising and comparing the performance of different simulations. This turns out to be very useful for system designers, since rather than just knowing the overall system performance, it helps to understand its evolution based on detailed information such as bandwidth usage or link saturation. As a consequence, if some

problems arise, it is easier to identify them, their possible causes and what is most valuable: which simulation parameter values perform best. For a detailed analysis of results comparing the BitTorrent protocol and our 2-LAMA approach reader s referred to [11].

Specifically, the following metrics are graphically displayed: (1) time required to share the data; (2) network consumed; (3) mean number of hops that traveling messages perform; (4) network channel usage; (5) network channel saturation; (6) number of cancelled messages; (7) cost associated to these cancels; and (8) source factor —measure that provides information about how data was actually distributed among peers. Next figure 7 shows time performance comparisons for different sharing methods (with and without norm adaptation) and different norm values ($\texttt{max}_{\texttt{BW}}$ in $normBW_{DL}$ and $\texttt{max}_{\texttt{Friends}}$ from $normFriends_{DL}$) [6].

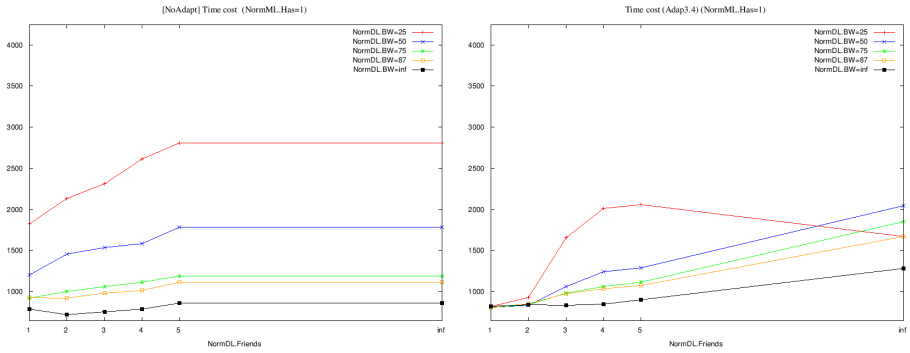

Fig. 7. Off-line comparison of different simulations

5 Conclusions and Future Work

This paper presents a simulator that has been developed with the aim of studying MAS organisational adaptation mechanisms in P2P scenarios. In order to endow the system with self-adaptation capabilities we advocate for adding a meta-level in charge of that task, instead of expecting participating agents to increase their behaviour complexity. The presented simulator provides the following functionalities and facilities:

- Definition and execution of simulations with different characteristics, as for instance network topology, number of peers. or sharing method.
- A GUI that permits to graphically control and follow simulations' evolution. Graphical representations are far more intuitive than textual logs, and if we

[6] In these plots $\texttt{max}_{\texttt{Has}}$ from $normHas_{ML}$ has been fixed to 1

also add the option to choose what to display (as in the user panel in our simulator), then the gain is even larger.
- Testing of different adaptation mechanisms for the social structure and norms.
- Process and analysis of simulation results. It generates different plots that help to compare the results of different simulations

Notice that while social adaptation is performed individually by each assistant within its cluster, in norm adaptation assistants have to reach an agreement on the norm new value. Specifically, each assistant computes its new desired norm values and later on they have to reach an agreement on each norm value. We believe that agreement technologies can play a key role in this process. Hence, our simulator can be used to test different approaches for reaching agreements among autonomous agents in the context of norm adaptation.

As future work, we plan to evaluate our approach with populations with norm violators, and in simulations where participants enter and leave. We are also interested in applying learning techniques to adaptation services.

Acknowledgements: This work is partially funded by IEA (TIN2006-15662-C02-01) and AT (CONSOLIDER CSD2007-0022) projects, EU-FEDER funds, the Catalan Gov. (Grant 2005-SGR-00093) and Marc Esteva's Ramon y Cajal contract.

References

1. Esteva, M.: Electronic Institutions: from specification to development. IIIA PhD. Vol. 19 (2003)
2. Hübner, J.F., Sichman, J.S., Boissier, O.: S-MOISE$^+$: A middleware for developing organised multi-agent systems. In: AAMAS Workshops. Volume 3913 of LNCS., Springer (2005) 64–78
3. Deloach, S.A., Oyenan, W.H., Matson, E.T.: A capabilities-based model for adaptive organizations. Autonomous Agents and Multi-Agent Systems **16**(1) (2008) 13–56
4. R., K., N., G., N., J.: Decentralised structural adaptation in agent organisations. In: AAMAS Workshop on Organised Adaptation in MAS. (2008)
5. Sims, M., Corkill, D., Lesser, V.: Automated Organization Design for Multi-agent Systems. Autonomous Agents and Multi-Agent Systems **16**(2) (2008) 151–185
6. Zhang, C., Abdallah, S., Lesser, V.: MASPA: Multi-Agent Automated Supervisory Policy Adaptation. Technical Report 03 (2008)
7. Campos, J., López-Sánchez, M., Esteva, M.: Multi-Agent System adaptation in a Peer-to-Peer scenario. In: ACM SAC09 - Agreement Technologies. (2009) 735–739
8. Campos, J., López-Sánchez, M., Esteva, M.: Assistance layer, a step forward in Multi-Agent Systems Coordination Support. In: International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2009) 1301–1302
9. Lewis, D.: Convention: A Philosophical Study. Harvard University Press (1969)
10. Cohen, B.: The BitTorrent Protocol Specification. http://www.bittorrent.org/beps/bep_0003.html
11. Campos, J., López-Sánchez, M., Esteva, M., Novo, A., Morales, J.: 2-LAMA Architecture vs. BitTorrent Protocol in a Peer-to-Peer Scenario. In: to appear in Twelfth Catalan Congress on Artificial Intelligence (CCIA09). (2009)