2-LAMA Architecture vs. BitTorrent Protocol in a Peer-to-Peer Scenario

Jordi CAMPOS [a&1], Maite LOPEZ-SANCHEZ [a], Marc ESTEVA [b], Alba NOVO [a] and Javier MORALES [b]

[a] *WAI, MAiA Dept., Universitat de Barcelona, Spain*
[b] *Artificial Intelligence Research Institute (IIIA) CSIC, Spain*

Abstract. In this paper, we review our Multi-Agent System (MAS) architecture (2-LAMA) proposed to assist existing MAS. This architecture consists of two levels: the conventional MAS system and an additional *meta-level* in charge of assistance. We illustrate our approach in a Peer-to-Peer sharing network scenario and compare it with the commonly used BitTorrent protocol. Experiments show that the cost of adding the *meta-layer* is lower than the obtained benefit. We conclude by claiming that our approach provides the system with the required flexibility to deal with dynamic environments.

Keywords. MAS, Organisation, Coordination

Introduction

Briefly, we can describe a Multi-Agent System (MAS) as a set of agents that interact within an environment to achieve their common and/or individual goals. Originally, Multi-agent Systems were designed *ad hoc* without any special methodology, developing their own infrastructure from scratch [4]. However, as MAS area evolved, certain tasks were abstracted and gradually provided by MAS infrastructure as domain independent services. Regarding agent interaction, it is structured through the *coordination model* (e.g. interaction protocols) and some of these services aid agents to enact it. Thus, we have proposed [1] the term *Coordination Support* to designate those infrastructure services that help to structure agent interactions. We group *Coordination Support* services in a generic set of layers that encompasses previous layering approaches [5,6,3]. First layers (*Connectivity*, *Agent Communication* and *Organisational* layers) are devoted to enable agents coordination at different levels, whereas the top layer (*Assistance Layer*) provides an added value by assisting coordination further than enabling it. This layer may even have pro-active capabilities that let the MAS infrastructure take the initiative and act intelligently —e.g. adapting previous layers depending on system's evolution. We propose two main focus for this purpose: assisting individual agents to follow the current *coordination model* and adapting this model to varying circumstances. In fact,

[1]Corresponding author: Facultat de Matemàtiques. Gran Via de les Corts Catalanes 585, 08007 Barcelona. e-mail: jcampos@maia.ub.es

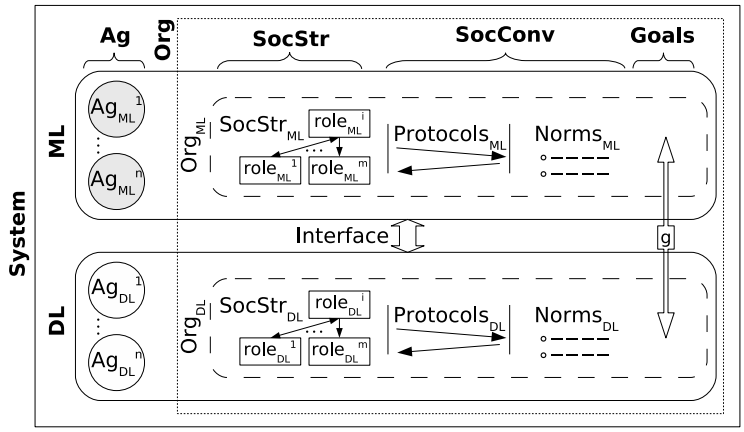

Figure 1. Two Level Assisted MAS Architecture (2-LAMA).

we identify assisting agents coordination –further than enabling it– as a potential area to enhance MAS and we expect that new research lines can arise.

This paper proposes a specific architecture to provide the *Assistance Layer* functionalities. We call it *Two Level Assisted MAS Architecture* (2-LAMA). Furthermore, we illustrate its deployment in a Peer-to-Peer sharing network scenario (P2P). This is a simplified scenario where *peers* (agents) share single pieces of data. The relationships they establish change over time depending on network status. Our vision is that these relationships define the system's organisation (i.e. how computers organise themselves to interact), whereas changes in network status constitute its dynamic environment. The performance of a system is computed in terms of time and network consumptions. This paper presents a comparison of the performance of this (2-LAMA) architecture with respect to a simplified version of the commonly used BitTorrent protocol[2].

This paper is structured as follows. First section is devoted to describe our proposed architecture from a general point of view. Afterwards, section two introduces our P2P context scenario and the BitTorrent protocol that we take as reference. Then, third section details how our 2-LAMA architecture can be applied to this P2P scenario and the changes in the protocol it entails. Fourth section illustrates this application with some experiments and their results. Finally, last section presents some conclusions and future work.

1. General Model: 2-LAMA

We propose a Two Level Assisted MAS Architecture (2-LAMA) that consists on adding a *meta-level* (ML) on top of a previously existing system we call *domain-level* (DL) plus an interface (Int) that communicates both levels (see Figure 1). Thus, our model can be expressed as: $M = \langle ML,\ DL,\ Int \rangle$. Each level has a set of agents (Ag_{xL} where xL is a generalisation of ML and DL) that are organised. We note its *Organisation* as Org_{xL}. Hence, each level can be defined as: $xL = \langle Ag_{xL},\ Org_{xL} \rangle$. Organisational main components are a social structure ($SocStr$), its social conventions ($SocConv$) and some organisational goals ($Goals$), thus, $Org = \langle SocStr, SocConv, Goals \rangle$.

The social structure consists of a set of roles (Rol) and the relationships (Rel) among agents playing them: $SocStr = \langle Rol,\ Rel \rangle$. In fact, role's possible actions are defined by social conventions. They may define valid sequences of actions and/or their consequences. The former is usually defined using protocols ($Prot$), and the latter is generally expressed by means of norms ($Norms$). Specifically, protocols define legitimate sequences of actions performed by agents playing certain roles. Whereas norms limit agent's actions and/or determine their consequences. In summary, $SocConv = \langle Prot,\ Norms \rangle$. Finally, organisational goals describe the proposal that guided the organisation design —which may differ from participant individual goals.

Furthermore, communication among levels covers bottom-up (Up) and top-down (Dn) information exchanges: $Int = \langle Up,\ Dn \rangle$. The *meta-level* perceives *domain-level* observable properties, evaluates them, and adapts *domain-level* organisation (Org'_{DL}). Perceived properties are those that can be observed in the environment ($EnvP$, e.g. date, temperature...) and those that can be observed in agents (AgP, e.g. colour, position...). Hence, $Up = \langle EnvP, AgP \rangle$ and $Dn = \langle Org'_{DL} \rangle$. On the other hand, we assume each *meta-level* agent ($a_{ML} \in Ag_{ML}$) has partial information about such properties, so it only perceives a subset of $EnvP$ and AgP (in many scenarios global information is not available). More concretely, each a_{ML} assists a given subset (i.e. a *cluster*) of *domain-level* agents. In this manner, each *meta-level* agent has partial information about its cluster and shares this information with other *meta-level* agents in order to better adapt the *domain-level* organisation.

In addition, in a specific domain, we can define a metric to evaluate a MAS performance —if the MAS has an organisation as described, its goals may be related to this metric. We use this domain-dependent performance metric to empirically evaluate our MAS proposal versus other approaches.

2. P2P Scenario

As a general illustration of the *Assistance Layer* and, in particular, the 2-LAMA Architecture, we present a P2P sharing network scenario. In such scenarios, a set of computers connected to the Internet (*peers*) share some data. Initially, not all of them have such data, but they exchange pieces of it in order to collect the whole information. In this work, for the sake of simplicity, we assume information is composed of a single piece of data. Furthermore, performance in this scenario is evaluated in terms of minimal time and network consumptions during the sharing process —in particular, we prioritise shorter times. Thus we can define as system's goal the minimisation of such measures, so that the faster the data is obtained and the less network bandwidth[2] is consumed, the better for the users. Notice, though, that there is a trade-off between time and network usage. Therefore, although a *peer* can potentially contact any other *peer*, it usually contacts just a subset in order to consume less network.

Nowadays BitTorrent is one of the most widely used protocols in P2P sharing network scenarios. Here we use it as a reference but simplify some features such as the

[2]This bandwidth is the capacity to transfer data over user's network connection. The less is used by the peer, the more is left for other purposes.

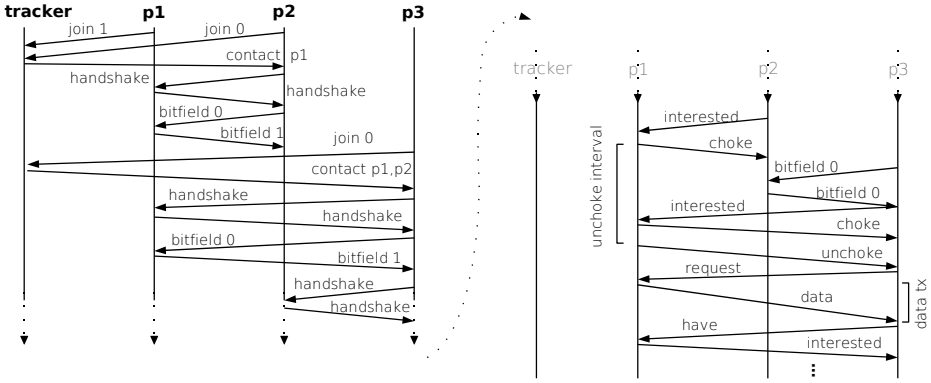

Figure 2. Initial stages of BitTorrent simplified protocol. p stands for *peer*.

multi-piece[3] data, which is taken to be a single-piece datum in our case. Figure 2 illustrates this simplified version. A *peer* gets involved in a sharing network by sending a "`join <hasDatum>`" message to a *tracker* —a software that tracks all *peers* that are currently sharing a certain data. "`<hasDatum>`=[1/0]" indicates if it has (1) or has not (0) the data —i.e. it is a *complete* or *uncompleted peer*. The *tracker* replies with a "`contact <peers>`" message, that contains a list of all current *peers*. Then, the *peer* exchanges "`handshake`" messages with all *peers* in that list. After that, it exchanges "`bitfield <hasDatum>`"[4] messages with the same *peers* to share their status.

After this handshake phase, *peers* lacking the data request it to the ones that have it by means of "`interested`" messages. As a response, all get "`choke`" messages, which mean that any further message will be ignored. Nevertheless, at certain time intervals (`unchoke_interval`), each *peer* having the data will send "`unchoke`" messages to some of the *peers* that were interested (*candidates*). The BitTorrent specification defines that four peers (`num_unchokes`) are selected among the *candidates*. Notice, though, that, due to the lack of space, $p1$ in Figure 2 is just *unchoking* one *peer* ($p3$). The selected *candidates* are those that were *choked* most recently. In case two of them were *choked* at the same time, the one having a larger network *bandwidth* (`upload_bw`[5]) is selected. In fact, if a *peer* interest is older than a defined interval ($\texttt{aging\_period} = \texttt{unchoke\_interval}/1.5$), its age is ignored and only its *peer*'s `upload_bw` is compared. In addition, in two out of three `unchoke_interval` selection processes, the fourth *peer* is randomly selected.

When a *peer* receives an "`unchoke`" message, if it does not have the data yet, it replies with a "`request`" message. In such case, the *complete peer* sends the data. Later, when the requester *peer* receives the data, it informs the *tracker* with a "`completed`" message. Next, it sends a "`have`" message to the other *uncompleted peers* to let

[3]Generally, P2P sharing networks split large data into small pieces. Then, *peers* collect these pieces from different sources to compose the whole data again.

[4]In the multi-piece BitTorrent, the "`bitfield`" message contains one bit for each piece to indicate if the *peer* has it or not.

[5]In a multi-piece scenario, this measure is estimated from previous piece interchanges. However, since in a single-piece implementation no estimation can be performed, its value is taken from the network definition.

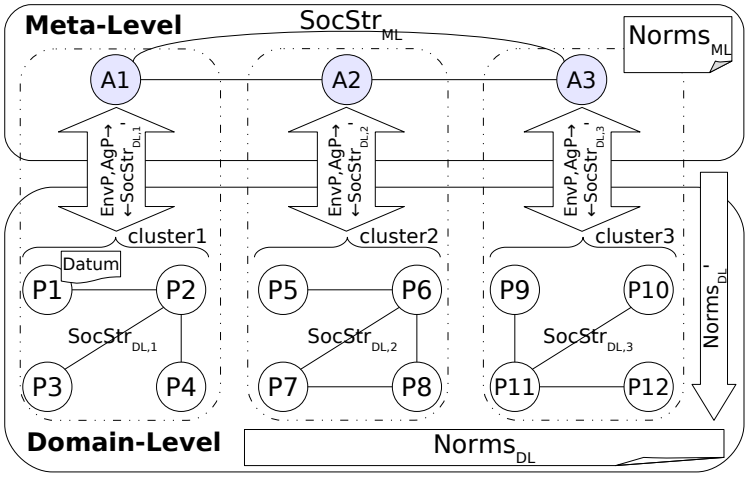

Figure 3. 2-LAMA in the P2P scenario.

them know that its status has changed. Then, some of them may probably send an "`interested`" message as described before.

3. The 2-LAMA Architecture Applied to P2P

This section is devoted to detail how the general model explained in previous section 1 can be applied to the specific Peer-to-Peer sharing network scenario. In order to do so, it first describes how agents are distributed in the general MAS architecture and afterwards details the protocol they follow to interact.

We model the P2P scenario as a MAS where *peers* sharing data are participant agents in the *domain-level* (Ag_{DL}). They play a single role $Rol_{DL} = \{peer\}$ within a certain organisation (Org_{DL}) (see Figure 3). We assume the organisational goal ($Goals$) is that all *peers* have the data consuming the minimal time and network.

As peers usually contact a subset of neighbours, we define it as the relationships among agents. These relationships, which belong to the social structure ($SocStr_{DL}$), will be updated by the *meta-level* taking into account the system status.

Regarding social conventions, *peers* use the protocol ($Prot_{DL}$) specified later on and two norms $Norm_{DL} = \{normBW_{DL}, normFriends_{DL}\}$. First norm ($normBW_{DL}$) 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". Second norm ($normFriends_{DL}$) limits the number of *peers* a *peer* can simultaneously send the data. Analogously to previous norm, we define $normFriends_{DL}$ ="a *peer* cannot simultaneously send the data to more than $\texttt{max}_{\texttt{Friends}}$ *peers*".

In order to provide assistance to the *domain-level*, we add the *meta-level* on top of it. This *meta-level* also has a single role $Rol_{ML} = \{assistant\}$. Each agent in Ag_{ML} assist a disjoint subset of *peers* (*cluster*$\subset Ag_{DL}$) It does it so by collecting information about them and adapting their local organisation. Its decisions are based on local information about its associated *cluster*, aggregated information about other *clusters* and the norms

at their level ($Norm_{ML}$). Some examples of local information are latencies ($EnvP$) or which *peers* have the data (AgP). Information about other *clusters* come from their neighbours in the *meta-level social structure ($SocStr_{ML}$)*. Regarding *meta-level* norms, we consider one that limits the number of *peers* –in the cluster– to inform about a new *peer* –in another cluster– having the data. Thus, when an assistant receives the information that one peer in another cluster has became completed, the number of peers it can decide to transmit this information to is limited. Therefore, the norm can be expressed as $normHas_{ML}$ ="Upon reception of a completed(peer $\notin$ $cluster$) message, inform no more than $\texttt{max}_{\texttt{Has}}$ *peers* $\in$ $cluster$ ". Finally, we assume *assistants* are located at Internet Service Providers (ISP) and thus related communications are fast[6].

We extend the BitTorrent simplified protocol in section 2 to include *meta-level* communications. Furthermore, we add an initial phase to estimate network latencies and vary the candidate selection criteria in order to exploit *meta-level* assistance.

In the new protocol, the *tracker* functionality is provided by *meta-level assistants*. Thus, a new *peer* sends its "`join`" message to the closest *assistant* —closest cluster in terms of network latencies. Then, the *assistant* asks the *peer* to measure its latencies with all other *peers* in its *cluster* by sending a "`get_latency <peers>`" message. The *peer* measures latencies by exchanging "`lat_req`"/"`lat_rpl`" messages, and informs back the *assistant* with a "`latency <amount>`" message. Once an *assistant* has all latencies among its *peers* ($EnvP$) and knows which ones have the datum, it estimates which would be the best organisation. Then it suggests the agent *relationships* (Rel'_{DL}) by sending "`contact <peers>`" messages to all the *peer*s in its *cluster*. Accordingly, in case a new agent enters the system, its *assistant* asks *this single peer* to measure its latencies against the rest of the *cluster*, and computes the best organisation again. On the contrary, if an agent leaves the system, its *assistant* can compute the new organisation without collecting new latencies. Notice that in current implementation there are no agents entering or leaving the system. In contrast to BitTorrent, in any case, the supplied list of *peers* does not include all *peers*, but only a subset of *peers* in its *cluster*. Afterwards, the previously introduced P2P protocol is followed by sending "`bitfield`" messages —i.e. "`handshake`" messages are omitted.

In contrast to BitTorrent protocol, there are no "`interested`" messages nor automatic "`choke`" replies. Instead, "`request`" messages are used and the data can be just sent. *Choke* replies only occur if the source peer is already serving more than $\texttt{max}_{\texttt{Friends}}$ *peers*. If this is the case, "`unchoke`" messages are sent as data transmissions end. On the other hand, a requester *peer* is allowed to get data from two sources simultaneously. This is done –for a short time– in order to compare their effective bandwidth so to choose the fastest source (the other one is discarded with a "`cancel`" message).

Finally, when a *peer* informs its *assistant* that it is *completed*, then this *assistant* informs its neighbour *assistants* with a "`completed_peer <peer>`" message. These *assistants* spread this information towards some of their *peers* –limited by $\texttt{max}_{\texttt{Has}}$– by means of a "`has_datum <peer>`" message. In that moment, those *peers* may request the datum to the new source. In addition, an *assistant* sends an "`all_completed`" message to its neighbour *assistants* when all its *peers* are completed.

[6]This approach is not unrealistic since, nowadays, there exist ISP initiatives [7] to improve –and be involved in– P2P distribution systems.

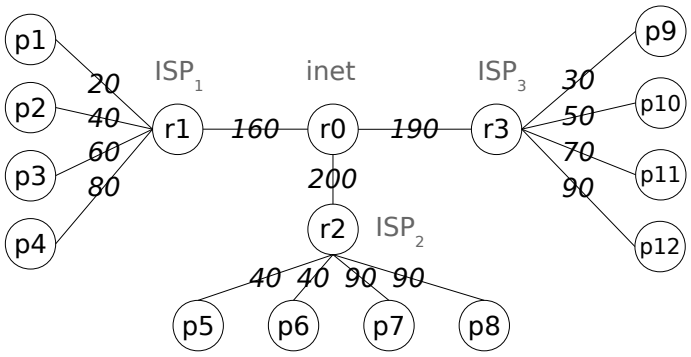

Figure 4. Network topology graph. Nodes represent *peers* (p) or *routers* (r) —among different *links* (the arcs). Arc weights correspond to link's *bandwidths*.

4. Experiments and Results

We have tested our approach on the P2P scenario described in section 2. In our experiments, we consider the network topology depicted in Figure 4. It consists of 12 *peers*[7] ($p1..p12$) connected through individual links to their corresponding Internet Service Provider $ISP_1..ISP_3$ ($r1..r3$ stand for routers). ISPs are in turn connected to the Internet ($r0$) through aggregated links, which are shared among the messages from their peers. As Figure 4 shows, links have different communication capacities. Overall, this topology provides us with a highly dynamic environment where communication latencies change depending on message traffic and channel sharing.

We have implemented a simulator in Repast Simphony that models agents and their message transport through the network. We used this simulator to test both BitTorrent and 2-LAMA approaches. In the BitTorrent configuration, a single *Tracker* is linked to $r0$. Whereas our 2-LAMA approach considers there is an *assistant* connected to each ISP ($r1..r3$). Each assistant is in charge of the set of peers (*cluster*) in an ISP —e.g. *assistant* a_1 is linked to $r1$ and it is in charge of $p1..p4$. In both approaches, these elements (*tracker*/*assistants*) have an infinite *bandwidth* (as if they were located at the router).

We define *bandwidth* as the number of data units that can traverse a channel in a time unit. Hence, the time required to transmit a message from one agent to another depends on: its length, the *bandwidths* of the traversed links, and the number of simultaneous messages traversing the same links —a link's *bandwidth* is divided among the messages that traverse it. In our simulations, we have used the following message lengths: `piece` messages require 5000 data units, `lat_req`/`lat_rpl` require 150 data units and all the other messages require a single data unit.

Regarding the configuration of our experiments, BitTorrent uses an `unchoke_interval` of 250 time units (ticks), which is approximately the time required to send four (i.e. $=$ `num_unchokes`) data messages along an average *individual link*. Accordingly, they use an `aging_period` of 130 ticks to keep the same ratio among these two constants than the one that is defined by the BitTorrent protocol. On the other hand, the 2-LAMA experiments have been performed with the following initial norm: `maxHas` $= \infty$,

[7] In this paper we are not dealing with entering/leaving *peers*.

	time	*cNet*	*nHops*	*nData*	*cLat*	*cML*
BT	933.3	206182	3.4	11	0	0
2-LAMA	811.1	316190	3.0	30.7	21600	6596.03
BT-P4	827	208650	3.5	11	0	0
2-LAMA-P4	667	285660	2.9	38	21600	6380

Table 1. Results from 2-LAMA and BitTorrent (averaged and a single configuration).

$\texttt{maxBW} = 100\%$, $\texttt{maxFriends} = 3$ —notice that last two norms are adapted at run-time by *meta-level*. These norms lead 2-LAMA approach to a similar initial behaviour than BitTorrent because: $\texttt{maxHas} = \infty$ does not restrict communications among *clusters*, $\texttt{maxBW} = 100\%$ does not limit *peer* communication and $\texttt{maxFriends} = 3$ is equivalent to the three non-random *unchoked peers*. In our current implementation, *domain-level* agents always fulfil norms[8].

Table 1 shows different evaluation metrics in BitTorrent (BT) and 2-LAMA approaches. We have tested both approaches by varying the *peer* that initially has the datum. Thus, tests include the average results for twelve different settings (so that they cover all possible initial data positions in a single *peer*). Furthermore, in order to exemplify a single execution, we have included the last two rows, which correspond to the execution with the data initially at *peer* P4, which is an intermediate case.

The main evaluation metric is the time required to spread the data among all peers (*time*). We also evaluate the *network cost* (*cNet*) consumed by all messages. The cost of a single message is the product of its length by the number of links it traverses. Thus, the table also shows the average number of links traversed by each message (*nHops*). In addition, the results include the number of data messages (*nData*), the cost of all `lat_req`/`lat_rpl` messages (*cLat*), and the cost of all messages related with the *meta-level*'s presence (*cML*) —i.e. all messages related with *assistants*.

If we compare the performance of both approaches (BT and 2-LAMA), we see that our proposal requires less time to share the data. This means that the time *peers* invest in communicating with our proposed *meta-level* and collecting the required information, is less than the benefits of having such an additional level. Even more, we expect larger differences in performance when repeating the data sharing process among the same *peers* since the information collected by our *meta-level* –e.g. measured latencies– will be used more than once. Thus, for example, In current 2-LAMA experiments, from 33 up to 56 ticks –depending on the *cluster* of *peers* – are invested in measuring latencies.

In contrast, the *network cost* (*cNet*) is larger in 2-LAMA. Our proposal requires more communication because it initially measures latencies (*cLat*), it has extra communications due to the *meta-level* (*cML*), and it sends more data messages (*nData*). Specifically, latency measurements (*cLat*) represent approximately a 20% of the *network cost* increment. This represents an initialisation phase that could be omitted in subsequent executions. On the other hand, in current scenario, 2-LAMA *peers* compare data sources by retrieving some data from them. This increases the number of data messages (*nData*) although most of them are cancelled. We expect to minimise this network consumption when dealing with more than one piece of data, since *peers* could compare sources depending on previous retrieved pieces. Regarding the number of links traversed

[8] Otherwise, we could count on an infrastructure mechanism at ISPs that detects and filters out messages that exceed the bandwidth limit (`maxBW`), or the simultaneous data messages limit ($\texttt{maxFriends} = 3$).

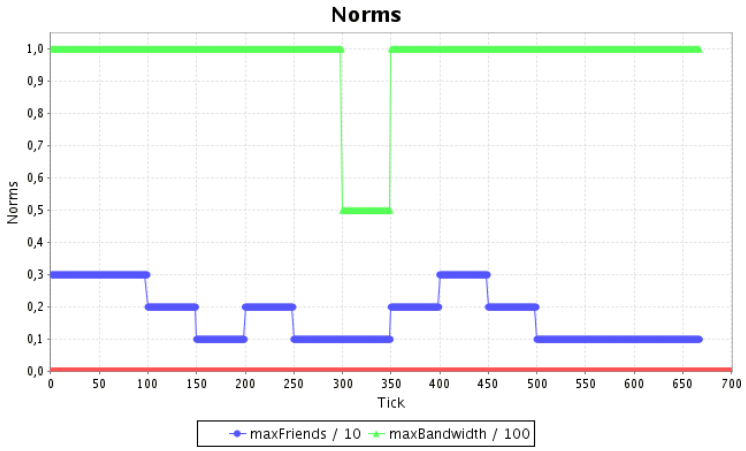

Figure 5. Norm adaptation during the a data sharing process.

by messages (*nHops*), our 2-LAMA approach has more local communications –i.e. intra-cluster– than BT. It is convenient because local messages have lower latencies and cost.

Finally, the empirical comparison conclusion depends on the importance of execution time versus network consumption. In particular, we assume time weights more than network. Thus, having a decrement of 10% in execution time is worth since the network increment does not saturate the communication channels yet. Under this criterion, we consider 2-LAMA is better than BT.

4.1. A norm adaptation case

Last two rows in table 1 compare two executions with the data initially at *peer* P4. In this case, the *meta-level* adapts *domain-level* norms as depicted in Figure 5. The $\texttt{max}_{\texttt{Friends}}$ limit starts with a value of 3 but decreases to 1 at the beginning, since there is only a single data source with a small *individual link* bandwidth. Next, it is increased to 2 to take more profit of the new data sources —i.e. the *peers* that obtained the data from the original source. However, this generates more network traffic on *aggregated links*, which saturates them and increases network latencies. Thus, the *meta-level* decreases $\texttt{max}_{\texttt{Friends}}$ again to 1 to reduce network traffic. Even $\texttt{max}_{\texttt{BW}}$ is decreased to 50% to reduce traffic. Once the data has been spread among *clusters*, it can be distributed without collapsing the *aggregated links*. Thus, both limits are increased again. Finally, when there are much more data sources than *peers* retrieving data, $\texttt{max}_{\texttt{Friends}}$ can decrease again, forcing a parallel distribution. From this we can see how 2-LAMA is able to re-organise itself depending on environment changes, with more flexibility than BT.

5. Conclusions

This paper presents our Two Layer Assisted MAS Architecture (2-LAMA), that adds a *meta-level* in charge of adapting the system to dynamic changes. The proposed adaptation is distributed requiring no global information. We illustrate our proposal by means of a P2P sharing network scenario. We also use it as a testbed for comparing our ap-

proach with the commonly used BitTorrent protocol. Experiments show interesting results, notably the fact that the cost of adding the *meta-layer* is lower than the obtained benefit. Hence, we conclude it is feasible and worth to add our proposed *meta-layer*.

As future work, we plan to further explore the adaptation mechanism by applying automated learning techniques. Moreover, we would like to deal with open MAS, where agents can join and leave and transgress social conventions. This would mean to deal with dynamical changes further than the dynamism due to sharing channels.

Acknowledgements. This work is partially funded by IEA (TIN2006-15662-C02-01), AT (CONSOLIDER CSD2007-0022) projects, and by Marc Esteva's Ramon y Cajal contract.

References

[1] Jordi Campos, Maite López-Sánchez, and Marc Esteva. Assistance layer, a step forward in Multi-Agent Systems Coordination Support. In *Autonomous Agents and Multiagent Systems*, pages 1301–1302, 2009.

[2] Bram Cohen. The BitTorrent Protocol Spec. http://www.bittorrent.org/beps/bep_0003.html.

[3] Foundation for Intelligent Physical Agents. *FIPA Abstract Architecture Specification*, December 2002. http://www.fipa.org/specs/fipa00001/.

[4] N. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

[5] A. Omicini, S. Ossowski, and A. Ricci. Coordination infrastructures in the engineering of multiagent systems. *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*, 11:273–296, 2004.

[6] Katia P. Sycara, Massimo Paolucci, Martin Van Velsen, and Joseph A. Giampapa. The RETSINA MAS infrastructure. *Autonomous Agents and Multi-Agent Systems*, 7(1-2):29–48, 2003.

[7] Haiyong Xie, Y. Richard Yang, Arvind Krishnamurthy, Yanbin Liu, and Avi Silberschatz. P4P: provider portal for applications. 2008.